

---

# **Interactive Power Flow**

**IPF Developers**

**Nov 07, 2022**



# CONTENTS

<b>1</b>	<b>Features</b>	<b>3</b>
<b>2</b>	<b>Contents</b>	<b>5</b>
2.1	Installation . . . . .	5
2.1.1	Docker . . . . .	5
2.1.2	Compiling Manually . . . . .	5
2.2	Overview . . . . .	6
2.2.1	IPF Interaction Model . . . . .	6
2.2.2	Executables . . . . .	9
2.2.3	Network Data . . . . .	10
2.2.4	The BASE (.bse) File . . . . .	13
2.3	History . . . . .	13
2.3.1	Original License . . . . .	14
2.3.2	Original Contributors . . . . .	15
2.4	Application Examples . . . . .	15
2.4.1	Introduction . . . . .	15
2.4.2	Setting Up a Network Data File . . . . .	15
2.4.3	New Facilities . . . . .	22
2.4.4	Reconductoring . . . . .	24
2.4.5	Series Compensation . . . . .	25
2.5	Record Formats . . . . .	26
2.5.1	Overview . . . . .	26
2.5.2	System Changes . . . . .	27
2.5.3	Comment . . . . .	29
2.5.4	Continuation Bus Data (+) . . . . .	29
2.5.5	Area Interchange Control (A) . . . . .	32
2.5.6	Area Output Sort (A0) . . . . .	33
2.5.7	AC Bus Data . . . . .	34
2.5.8	AC Bus Data (B-blank) . . . . .	37
2.5.9	AC Bus Data (BC) . . . . .	40
2.5.10	AC Bus Data (BE) . . . . .	42
2.5.11	AC Bus Data (BF) . . . . .	44
2.5.12	AC Bus Data (BG) . . . . .	44
2.5.13	AC Bus Data (BQ) . . . . .	47
2.5.14	AC Bus Data (BS) . . . . .	48
2.5.15	AC Bus Data (BT) . . . . .	51
2.5.16	AC Bus Data (BV) . . . . .	52
2.5.17	AC Bus Data (BX) . . . . .	53
2.5.18	Two-Terminal DC Bus (BD) . . . . .	57
2.5.19	Multi-Terminal DC Bus (BM) . . . . .	58

2.5.20	Delete Buses by Area (DA)	61
2.5.21	Delete Buses by Zone (DZ)	61
2.5.22	Equivalent Transmission Line Branch (E)	62
2.5.23	Scheduled Area Intertie (I)	65
2.5.24	Balanced Transmission Line Branch (L)	66
2.5.25	Two-Terminal DC Line (LD)	69
2.5.26	Multiterminal DC Line (LM)	70
2.5.27	Factor Change (PO, PZ, PN, PA, PB, PC, PD)	73
2.5.28	Reactive Capability Curves (QP, QX, QN)	78
2.5.29	Regulating Transformer (R, RV, RQ, RP, RN, RM)	81
2.5.30	Series Compensated RANI Model (RZ)	84
2.5.31	Transformer Data (T, TP)	85
2.5.32	Switched Reactance (X)	89
2.5.33	Zone Rename (Z)	91
2.5.34	Area Rename (ZA)	92
2.5.35	Bus Rename (ZB)	93
2.6	Power Flow Control (PFC)	94
2.6.1	Overview	94
2.6.2	The bpf Control Language	94
2.6.3	Special Characters	96
2.6.4	Default Convention	96
2.6.5	Microfiche Control Statement	96
2.6.6	Level 1 PFC Commands	97
2.6.7	Level 2 and 3 PFC Commands	97
2.6.8	PFC Commands	98
2.6.9	AGC	99
2.6.10	AI_LIST	100
2.6.11	ANALYSIS_SELECT	100
2.6.12	BRANCH_DATA	102
2.6.13	BUS_SENSITIVITIES	102
2.6.14	CHANGE_BUS_TYPE	105
2.6.15	CHANGE_PARAMETERS	109
2.6.16	CHANGES	114
2.6.17	COMMENT	114
2.6.18	COMMON_MODE_ANALYSIS	114
2.6.19	F_ANALYSIS_RPT	115
2.6.20	F_INPUT_LIST	116
2.6.21	F_OUTPUT_LIST	117
2.6.22	GEN_DROP	117
2.6.23	HEADER	120
2.6.24	INCLUDE_CONTROL	120
2.6.25	LINE_EFF	121
2.6.26	LINE_SENSITIVITIES	121
2.6.27	%LOAD_DISTRIBUTION	123
2.6.28	LOAD_GE	130
2.6.29	LOAD_PTI	131
2.6.30	LOSS_SENSITIVITIES	131
2.6.31	MERGE_OLD_BASE and MERGE_NEW_BASE	133
2.6.32	MVA_BASE	136
2.6.33	NETWORK_DATA	137
2.6.34	NEW_BASE	137
2.6.35	OI_LIST	137
2.6.36	OLD_BASE	141
2.6.37	OUTAGE_SIMULATION	142



2.6.38	OVERLOAD_RPT	146
2.6.39	P_ANALYSIS_RPT	146
2.6.40	P_INPUT_LIST	148
2.6.41	P_OUTPUT_LIST	149
2.6.42	REBUILD	149
2.6.43	REDUCTION	149
2.6.44	RPT_SORT	153
2.6.45	SAVE_FILE	153
2.6.46	SOLUTION	154
2.6.47	SORT_ANALYSIS	158
2.6.48	TRACE	158
2.6.49	TRANSFER_SENSITIVITIES	158
2.6.50	TX_EFF	159
2.6.51	USER_ANALYSIS	159
2.6.52	PFC Examples	169
2.7	Powerflow Command Language (PCL)	174
2.7.1	Introduction	174
2.7.2	General	176
2.7.3	File Opening and Saving	177
2.7.4	Processes	179
2.7.5	PUT_DATA	197
2.7.6	Report Generation	197
2.8	Command Line Tools	205
2.8.1	bpf	205
2.8.2	ipf_reports	207
2.8.3	ipfplot	209
2.8.4	netdat	209
2.8.5	ipfcut	209
2.8.6	pvcurve	212
2.8.7	post_pvcurve	212
2.8.8	qvcurve	212
2.8.9	findout	212
2.8.10	lineflow	213
2.8.11	mimic	213
2.8.12	ipfsrv	213
2.8.13	ipfbat	213
2.8.14	ipf_test	215
2.8.15	ipfnet	215
2.8.16	ips2ipf	218
2.9	Transient Stability Program (tsp)	221
2.9.1	Program Description	221
2.10	X Window Graphical Interface (gui)	222
2.10.1	Audience	223
2.10.2	User Interface	223
2.10.3	A Summary of Motif Basics	226
2.10.4	IPF as an X Client	229
2.10.5	IPF X Window GUI Architecture	231
2.10.6	History	231
2.10.7	Working with the GUI	232
2.10.8	Starting IPF	233
2.10.9	Exiting IPF	236
2.10.10	Opening Files	236
2.10.11	Saving Files	237
2.10.12	Changing the Displayed Network Size	237

2.10.13	Editing Base Case Data . . . . .	239
2.10.14	Solving a Network Case . . . . .	245
2.10.15	Bus and Branch Editing . . . . .	246
2.10.16	Input Data Edit Dialog Boxes . . . . .	247
2.10.17	AC Bus Input Data Boxes . . . . .	248
2.10.18	BD Bus . . . . .	252
2.10.19	BM Bus . . . . .	254
2.10.20	Continuation Bus . . . . .	254
2.10.21	Switched Reactance . . . . .	257
2.10.22	PQ Curve . . . . .	257
2.10.23	Sectionalization . . . . .	260
2.10.24	Line Tapping (may not be available) . . . . .	261
2.10.25	Transmission Line . . . . .	263
2.10.26	Phase Shifter . . . . .	265
2.10.27	Transformer . . . . .	267
2.10.28	Regulating Transformer . . . . .	269
2.10.29	Equivalent Network . . . . .	269
2.10.30	Menu Commands . . . . .	272
2.10.31	Customizing the GUI (XGUI) . . . . .	314
2.11	CFLOW C API (libcflow) . . . . .	319
2.11.1	Overview . . . . .	319
2.11.2	Creating a Program . . . . .	320
2.11.3	Running a CFLOW Program . . . . .	321
2.11.4	Debugging . . . . .	322
2.11.5	Functions Overview . . . . .	323
2.11.6	Notes . . . . .	325
2.11.7	Simple Report Example . . . . .	327
2.11.8	Standard Line Flow Summary Example . . . . .	329
2.11.9	incrm Program . . . . .	333
2.11.10	libcflow API Reference . . . . .	348
2.12	Network Diagrams . . . . .	416
2.12.1	Overview . . . . .	416
2.12.2	Input Requirements, Output, and Operation . . . . .	416
2.12.3	Coordinate File . . . . .	417
2.12.4	PostScript Procedures . . . . .	428
2.12.5	Diagram Components . . . . .	432
2.13	Calculating Line Impedance . . . . .	438
2.13.1	Description of Conductor Data Fields . . . . .	438
2.13.2	Output Values . . . . .	440
2.13.3	Calculating the Impedance . . . . .	443
2.14	DC Line Modeling . . . . .	443
2.15	Network Reduction . . . . .	446
2.15.1	Method of Reduction . . . . .	446
2.15.2	Description of Reduction . . . . .	451
2.15.3	Program Control Options . . . . .	451
2.15.4	REI Equivalent . . . . .	454
2.15.5	REI Clusters . . . . .	456
2.15.6	Coherency Clusters . . . . .	457
2.15.7	Using REI Equivalents . . . . .	458
2.15.8	Optimal Network Determination . . . . .	458
2.15.9	REDUCTION Command . . . . .	458
2.16	Retained Network Definition . . . . .	458
2.16.1	Reduction Cards . . . . .	458
2.16.2	Input listings . . . . .	460

2.17	Voltage Limits and Starting Voltages . . . . .	460
2.17.1	Notes . . . . .	461
2.18	Developer Notes . . . . .	462
2.18.1	Debugging . . . . .	462
2.18.2	bpf Flowchart . . . . .	462
2.18.3	tsp Flowchart . . . . .	462
2.18.4	Variables . . . . .	462
2.18.5	Command Subroutines . . . . .	465
2.18.6	WSCC Specifications . . . . .	471
2.18.7	Additional Resources . . . . .	472
<b>Index</b>		<b>473</b>



Interactive Power Flow (IPF) is a software package for doing power flow studies. IPF models the operation of a bulk electric power network. It can be used to:

- Investigate bulk electric power networks
- Determine bus voltage distribution
- Study real and reactive power flows in lines
- Evaluate line overloads
- Plan area interchange control
- Determine transformer tap settings
- Determine remote-bus voltage controls
- Plan for system reactive power requirements
- Determine effect of load shedding, generator dropping, and line outages
- Run transient stability studies on generator rotor dynamics

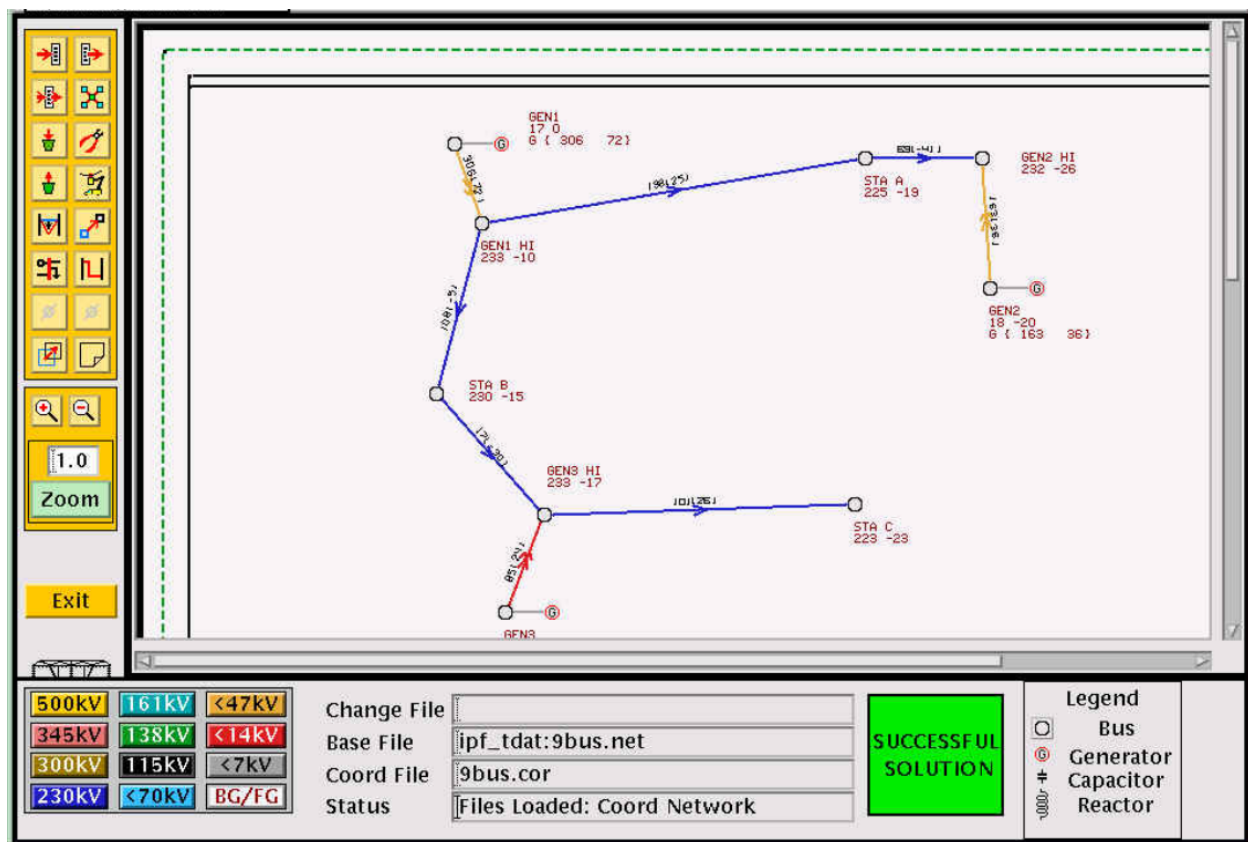


Fig. 1: Interactive Power Flow X Window GUI

Electric power system network design encompasses the following tasks:

- Determination of load centers and generation patterns as well as sizes of loads and generation.
- Determination of available transmission corridors (rights-of-way) and assessment of the capacity of these corridors to accommodate transmission lines.

- Evaluation of existing or planned networks with regard to adequate power-carrying capability, voltage regulation, reliability of service, and operating economics.
- Determination of size and routing of new transmission lines, and size and location of terminal equipment for achieving efficient and economical reinforcements when needed.
- Evaluation of proposed reinforcements in light of power flow capability, ability to withstand transient disturbances, reliability of overall service, economics, impact on regional economy, environment, energy conservation and operational constraints such as construction lead times, coordination of various facility ownership interests, flexibility for future growth and compatibility with other long-range plans.

The dynamic nature of load growth, load distribution, and generation patterns make the problem of network design one of planning. To plan for the future, power system planners and design engineers must look at the past and present. This makes Interactive Power Flow a great tool for the network design engineer, but it can also be used by students for academic studies. IPF models the complex network structure and can evaluate it at various points in time.

## FEATURES

Interactive Power Flow has the following high level feature set:

- Set of *Command Line Tools* for running power flow studies using domain specific languages (See *Powerflow Command Language (PCL)* and *Power Flow Control (PFC)* commands) for “batch” command line runs in which users can run multiple powerflow scenarios, plot results (See *ipfplot*, and create reports (See *ipf\_reports*)
- *X Window Graphical Interface (gui)* allowing easy-to-edit graphical display of the network diagram
- *CFLOW C API (libcflow)* functions for running studies
- Advanced capabilities like REI *Network Reduction* and *Transient Stability Program (tsp)*





## CONTENTS

## 2.1 Installation

IPF is installed by compiling from source. This is probably easiest using the Docker approach outlined below, but compiling manually from source is also an option.

### 2.1.1 Docker

You can use docker to build and run this project. Just build the container:

```
$ git clone https://github.com/mbheinen/bpa-ipf-tsp
$ cd bpa-ipf-tsp
$ docker build -t bpa-ipf-tsp .
```

Then run it using interactive `-i` and `tty -t` option. The container will run `bpf` by default so once the container is started you can just type the name of the file in the data directory that you'd like to run (e.g. `bench.pfc`):

```
$ docker run -it --rm bpa-ipf-tsp
BPA POWER FLOW PROGRAM VERSION:IPF 327
Enter Power Flow Control (PFC) file name > bench.pfc
```

Otherwise start an interactive `tty` with `bash` to run other processes.

```
$ docker run -it --rm bpa-ipf-tsp bash
[root@e9d28ac4f070 data]# bpf bench.pfc
[root@e9d28ac4f070 data]# tsp bench.fil
```

### 2.1.2 Compiling Manually

If Docker isn't an option, compile manually. The majority of this codebase is written in Fortran with some C so both Fortran and C compiler are needed in order to compile it. Also, note that to this point it has only been test compiled on CentOS/RHEL 7 (gcc 4.8.5) and Ubuntu 20.04 (gcc 9.4.0). Pull Requests are welcome if you want to try a different platform or run into issues on another platform and want to fix something!

Install the build tool ([CMake](#)), compilers, and dependencies ([Motif X Window](#)):

#### CentOS/RHEL

```
$ yum install cmake gcc gcc-gfortran motif motif-devel
```

#### Ubuntu

```
$ apt-get install cmake gcc gfortran libmotif-dev libxmu-dev
```

Then you're ready to compile. This project uses [CMake](#). CMake is a multi-platform build tool that can generate build files for many different target platforms. CMake recommends doing “out of source” builds, that is, the build files and artifacts are separated from the source files. This is convenient when doing development because there is no need to clean out compiled stuff (e.g. object files, libraries, executables, etc.) from the source tree. To do this, you create a `build/` directory at the top level of the project and everything gets built there. This allows you to just delete the `build/` directory when you're done.

Doing a checkout and compile of this repository is done as follows:

```
$ git clone https://github.com/mbheinen/bpa-ipf-tsp
$ cd bpa-ipf-tsp
$ mkdir build
$ cd build
$ cmake ..
$ make
$ sudo make install
```

After running these commands, you will see the library binaries in a `lib/` directory and the executables in a `bin/` directory. The `sudo make install` command will also place the binaries in `/usr/local/bin`, which should be in your path.

### Debug

To build with debug symbols, do `cmake -DCMAKE_BUILD_TYPE=Debug ..` instead of `cmake ..`

### Testing

You can run the test suite after compiling by running `ctest -C <Build Type>`:`

```
$ ctest -C Release
```

## 2.2 Overview

You can interact with IPF using the *Command Line Tools* (`bpf`, `ipfbat`, `ipfcut`, etc.), the *X Window Graphical Interface (gui)* or the *CFLOW C API (libcfw)*. Many of the functions and features of the command line tools are available through specially designed GUI features. The GUI simplifies creating network data, running the base case solutions, and graphically visualizing the network diagrams.

### 2.2.1 IPF Interaction Model

The conceptual model of IPF is quite simple. You load power system network data into IPF; the IPF solution “engine” performs the calculations for the solution, and then outputs this solution data.

IPF offers different approaches to accomplish power system solutions. Their style of interaction and processing are quite different.

- The batch power flow (e.g. *bpf* or *ipfnet*) approach. This is an Input-Process-Output approach. You write a command file containing all of the “orders” that you want filled, and the program performs the actions it determines are necessary to produce the ordered results.

- The Graphical User Interface (GUI) approach. This is command oriented - you click a button or enter a command, and it is executed immediately by the backend power flow engine (`ipfsrv`).
- The CFLOW approach (the C API called `libcflow`). This is a programming based approach where you write C code to interact with IPF.

Two Domain Specific Languages (DSLs) called Powerflow Command Language (PCL) and Power Flow Control (PFC) allow users to define the “scenarios” or “orders”. PCL is directly available through a Command Dialog window in the X Window GUI and can be used with the `ipfbat` program. PFC is used with the `bpf` program. See [Power Flow Control \(PFC\)](#) for details of the PFC syntax and [Powerflow Command Language \(PCL\)](#) for details of the PCL syntax.

---

**Note:** The PFC syntax was developed first and later [Bonneville Power Administration \(BPA\)](#) added the PCL syntax. They referred to PCL as the “new style”, and the PFC as the “old style”. These two sets of commands are not completely compatible even though the “new style” command set and syntax is closely modeled on the “old style.”

---

### The `bpf` Batch Approach

When you use `bpf`, you must first create a [Powerflow Command Language \(PCL\)](#) file with the appropriate commands to accomplish the solution task at hand. At runtime these commands are accepted by `“bpf”` and executed according to a logical processing order determined by the program. Hence you need not be concerned with the ordering of commands in your PFC file. Input commands will be processed first, and a solution done automatically before any output is produced. Finally, a new base file will be created, if you have requested one. See [PFC Examples](#) for examples of PFC files.

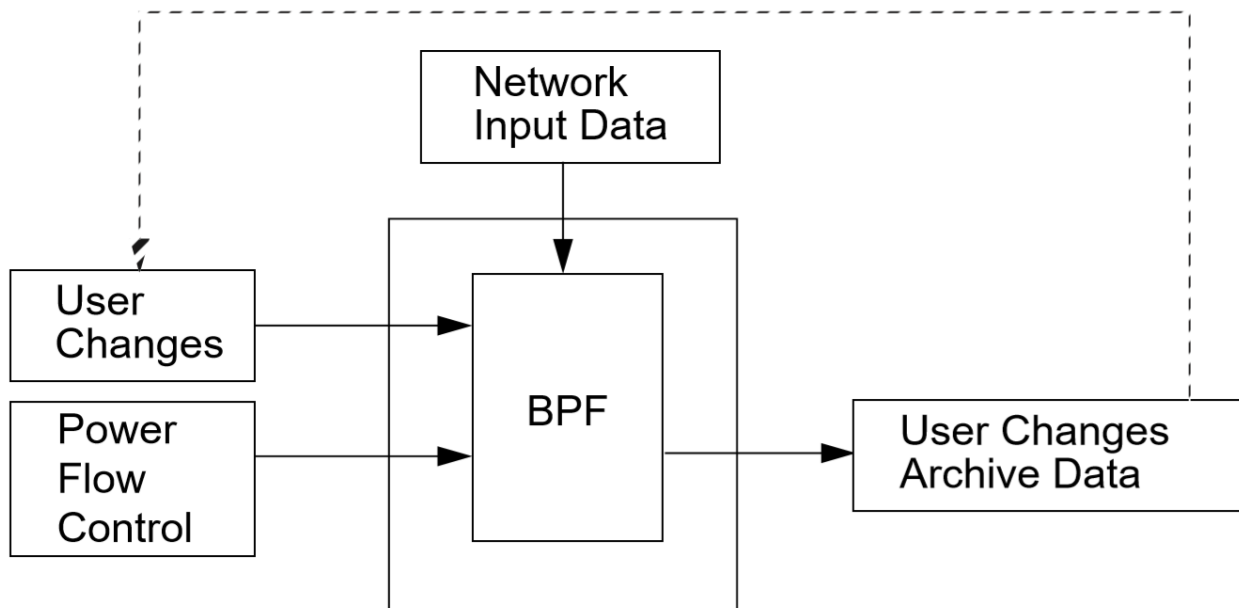


Fig. 2.2.1: BPF Information Flow Model

## The ipfbat Batch Approach

`ipfbat` allows you to interact with the solution “engine” (`ipfsrv`) without the GUI. You first create a PCL file with the appropriate commands, in the right order, to accomplish the solution task at hand. At runtime these commands are interpreted by `ipfbat`. The PCL file commands are processed sequentially. Additional PCL command files may be specified by name, so that a “chain” of PCL files may be processed in one run.

## The GUI Approach

When you use the GUI approach, you use an X Window graphical interface with dialog boxes, menus, windows, etc. This makes data input, output, and manipulation easy. In addition to allowing basic case solution tasks to be accomplished, certain specialized tasks such as line impedance calculations are available. See [Calculating Line Impedance](#) for details. However, for more involved tasks, you’ll need to use the `bpf` approach. For information about how to work with the GUI dialog boxes, menus, windows, etc., see [X Window Graphical Interface \(gui\)](#). That section also has a tutorial to show you how to solve straightforward power system cases.

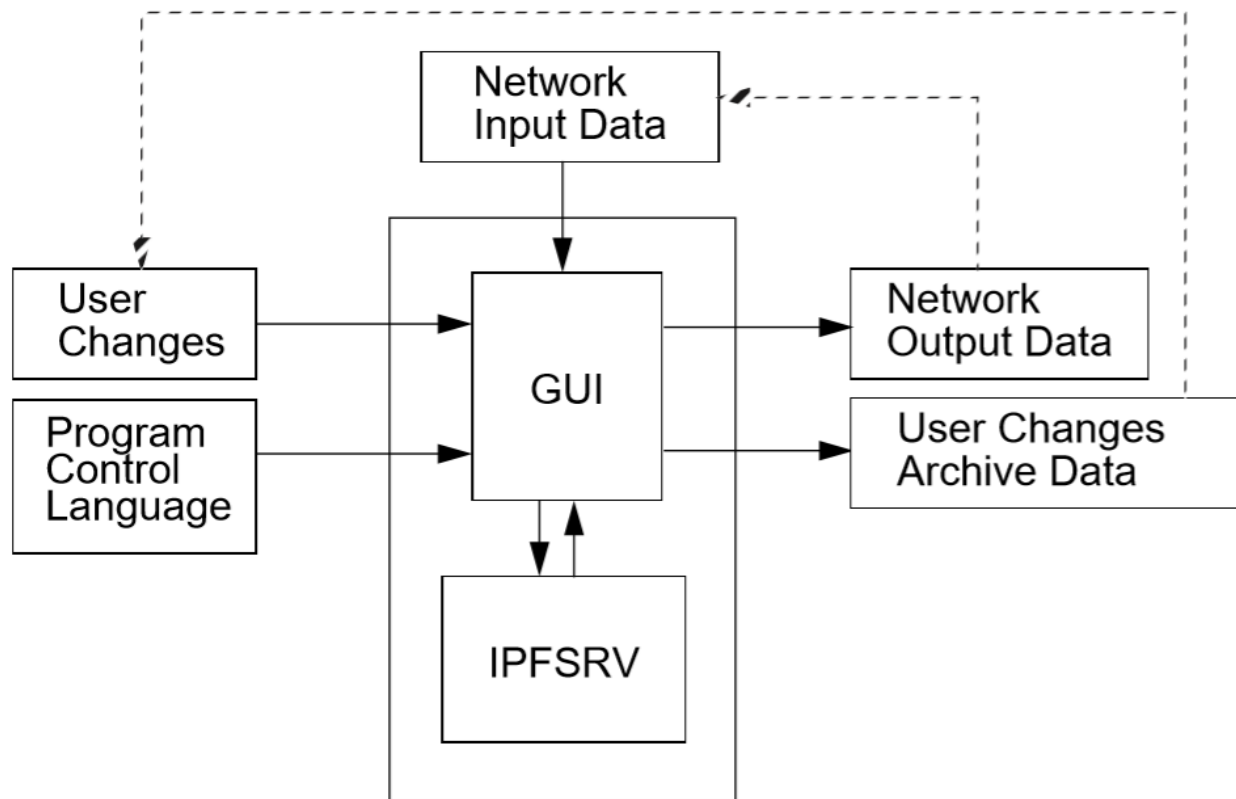


Fig. 2.2.2: PCL Information Flow Model

The core powerflow engine (`ipfsrv`) can be ran on separate machine from the graphical user interface. These two (`gui` and `ipfsrv`) processes are joined by interprocess communication (IPC) routines that shuttle data and instruction messages back and forth between the GUI and powerflow program using sockets. `ipfsrv` serves primarily as a solution and data engine that sends and receives data when requested by the user through the GUI.

## The CFLOW Approach

Many times users need to do a large number of similar runs or they need to process data from another system or set of files before running studies. CFLOW is a C library API (*libcflow*) for IPF. To use CFLOW, you write a C program, including the header file *cflowlib.h*, which defines all the structures, unions, and functions which allow access to the powerflow input and solution. To retrieve solution values, you call various CFLOW functions. The API interacts with *ipfsrv* internally. This allows you to do things like ask for a new solution, change the model, etc. See *CFLOW C API (libcflow)* for details on the API and examples.

## 2.2.2 Executables

IPF consists of many executable programs. Several of the most common executables are briefly described below.

### *bpf*

Command line program that performs power flow. *bpf* is the batch form of the powerflow program and unlike *ipfsrv* it doesn't use socket communication to interact with the powerflow engine. Rather it uses direct code library linking. It allows one or more scenarios to be defined in input files and run all at once via a call to *bpf* from a command line terminal. It executes using the commands from a *Power Flow Control (PFC)* file. Example usage: *bpf bench.pfc*. The PFC commands (i.e. the content of the .pfc file) used with *bpf* allow for complete power flow runs including defining the network model and commands to perform various operations. The *Record Formats* section describes the network model records available and the *Power Flow Control (PFC)* section describes the PFC syntax and commands available. More documentation about this program is in *bpf* section.

### *gui*

Launches the Graphical User Interface program built with **'Motif X Window'** that works in conjunction with the power flow server, *ipfsrv*. When the editing and displaying of buses and branches is being handled by the *gui* process, the work of calculating solution voltages for a given power system network is done by *ipfsrv*, which is just the *bpf* program in a different guise. More documentation about this program is in *X Window Graphical Interface (gui)* section.

### *ipfsrv*

The power flow service daemon which runs as the backend component of the *gui*. It executes *Powerflow Command Language (PCL)* commands through scripts dispatched from the *gui*. See *ipfsrv* for more details.

### *ipfbat*

Command line program that is the batch version of *ipfsrv*. It accepts a Powerflow Command Language (.pcl) file. Plotting can be done with a control file; however, for most plots *ipfplot* is easier to use. Example of use: *ipfbat bench.pcl*. The PCL commands used with *ipfsrv* and *ipfbat* are described in *Powerflow Command Language (PCL)*.

### *ipfcut*

Command line program that cuts out a subsystem from a solved base case file (.bse). Flows at the cut branches are converted into equivalent generation or load on specially formatted +A continuation bus records (read more about *Continuation Bus Data (+)*). An ensuing power flow run should solve with internal branch flows and bus voltages which are identical to those quantities in the original base case. More documentation about this program is in *ipfcut* section. Several methods are available to define the cut system: bus names, zones, base kVs, and individual branches. A pi-back feature replaces selected buses with a passive-node sequence (lines consisting of sections) with the original loads pi-backed in proportion to the line admittances.

### ipfplot

Command line plotting program to produce printed maps in batch. The program accepts a coordinate file (.cor) and a base case file (.bse) on the command line, as well as an optional second base case file. When the second base case file is specified, a difference plot is produced. You can also use `ipfplot` to produce bubble diagrams. The same coordinate files are used for both `gui` and `ipfplot`, but not all capabilities are available in `gui`. Documentation is in [Network Diagrams](#). More documentation about this program is in [ipfplot](#) section.

### tsp

Command line program that performs transient stability studies. Users specify power system network disturbances (line faults, generator trips), and the program simulates the affects on power system dynamics including key data points like generator rotor angle. This tool can be used to evaluate short term (on order of seconds to minutes) affects of these disturbances. Example of use: `tsp bench.fil`. More documentation about this program is in [Transient Stability Program \(tsp\)](#) section.

### ipfnet

The command line batch version of the “save netdata file” command built into the `gui / ipfsrv`. This program generates a WSCC-formatted network data file in any of the following dialects: BPA, WSCC, or PTI. “Dialects” means that the file is still WSCC, but the data is generated with special processing or restrictions and is destined for use with other programs. In the case of the PTI dialect, that data is preprocessed by the PTI-proprietary conversion program WSCFOR. Detailed documentation is in [ipfnet](#).

### ipf\_reports

The command line report tool to create output reports and summaries from solved base cases (.bse). Enter `ipf_reports` from the command line and follow the prompts.

### ips2ipf

The program that converts a network data file from WSCC’s Interactive Powerflow System (IPS) format to IPF format. Duplicate buses are renamed; Load Tap Changer (LTC) steps are converted to taps, shunt susceptance on slack and BQ buses are transferred to +A records; sectionalized lines containing a section 0 are renumbered 1, 2, ... ; BX, X, and remote controlled bus data are converted to IPF format, etc. Documentation is in [IPS IPF Differences](#).

## 2.2.3 Network Data

A core component to any power system analysis problem is the power system network (the interconnected lines, generators, loads, transformers, nodes/buses, breakers, and other components). In IPF, this is referred to as Network Data. Network Data defines the structure (connectivity) and properties (base voltage, real power injections, real and reactive power loads, transformer tap settings, etc.) of the power system network. This data is defined in files that contain bus (nodes) and branch (lines, transformers, etc.) records. Most of the input files are ASCII text files. But one important file, the base case file (.bse), is in binary format. You can, of course, edit ASCII text files with any text editor. This is often done, but your data integrity is safer if you do all the editing you can in the GUI. The base case binary file cannot be edited with an ordinary text editor, but can in effect be edited via the GUI when you have a base case file loaded.

There are multiple methods for specifying the Network Data. The following is a list of the various methods with descriptions.

### .pfc

This input file contains job control information for the `bpf` program. This file may contain Network Data explicitly (listing out [Record Formats](#) right in the file), but more often includes a property called `NETWORK_DATA` that specifies a file containing the [Record Formats](#) in ASCII format. Alternatively, users

can specify a property called OLD\_BASE that specifies a base case (.bse) file, or other job control data to be described. Keeping the Network Data out of the .pfc file allows more clear separation between the commands of the batch study and the network model.

You can edit this file using any ASCII text editor to add, modify, and delete commands and data records. See [PFC Examples](#) for examples.

## NETWORK\_DATA

This ASCII text input file consists of area, bus, and branch records. It must not contain modification records.

This file can be maintained by using an ASCII text editor. Or you can edit the records you want in the GUI through the various dialog boxes and then save a new NETWORK\_DATA file. In the file, data records may be in random order, but actual processing is done in the following order:

1. Area interchange records.

Each area record identifies a composition of zones whose member (associated) buses define specific aggregate quantities that may be controlled to specified export values.

A (Area interchange records)

I (Area intertie records)

2. Bus data record group containing at least two records.

Each bus data record identifies one bus in the network. Buses are uniquely identified by their bus name and base kV.

B (Bus records)

+ (Continuation bus records)

X (Switched Reactance records)

Q (PQ Curve data records)

3. Branch data record group containing at least one record.

L (AC or DC Transmission line records)

E (Equivalent Branch records)

T (Transformer records)

R (Regulators (Automatic or LTC transformer) records)

Branch data entered in any of the ASCII files is *single-entry* or one-way only. This means, for example, that a branch connecting buses A and B has a user-submitted entry (A,B) or (B,A) but not both. The program transposes the record internally as required during execution. Normally which way the branch is entered does not matter, but it does affect the default end metered on a tie line, and the physical position of line sections. See [Record Formats](#), for a discussion of this feature.

Branches are uniquely identified by three fields:

- Their terminal bus names and base kVs.
- Their circuit or parallel ID code.
- Their section code.

## BRANCH\_DATA

This ASCII text input file contains the branch data of all branches coded with in-service date and out-of-service date. This file is searched for branches in service on the date requested. `bpf` selects the appropriate branches.

### NEW\_BASE

This program-generated, binary output file contains complete base network data and steady-state operating values for the case being processed. This file is identical in format to the `OLD_BASE` file. `NEW_BASE` simply designates the file when it is produced as the output from a recently concluded case study. See [The BASE \(.bse\) File](#) section for more details on this file.

### OLD\_BASE

This program-generated, binary input file contains complete base network data and steady-state operating values. This file is identical in format to the `NEW_BASE` file. `OLD_BASE` simply designates the file when it functions as an already existing input file. See [The BASE \(.bse\) File](#) section for more details on this file.

### CHANGE

This ASCII text input file contains changes (new and modification records) to the data input from any combination of `NETWORK_DATA`, `BRANCH_DATA`, and `OLD_BASE` files making up the case to be studied. These change records change the input data for the base case.

### Printout File .pfo

This is an ASCII text output file that contains bus, branch, and solution data from a completed case study and is intended for ordinary, paper hardcopy output.

### Microfiche file .pff

This is a special format output file that contains bus, branch, and solution data from a completed case study and is intended for microfiche format.

Table 2.2.1: IPF Input/Output Files

File	Format	Input/Output (I/O)	Created by	Editing	Information Contained
<code>.pfc</code>	ASCII	<code>bpf</code> (I)	User	Yes	Bus, Branch, Commands, File Names
<code>.pcl</code>	ASCII	<code>gui</code> , <code>ipbat</code> (I)	User	Yes	Commands, File Names
<code>NET-WORK_DATA</code>	ASCII	<code>bpf</code> (I) <code>GUI</code> , <code>IPFBAT</code> (I/O)	User <code>gui</code> <code>ipfnet</code>	Yes	Bus, Branch
<code>BRANCH_DATA</code>	ASCII	Input Only	User	Yes	Branch
<code>OLD_BASE .bse</code>	Binary	Input Only	IPF	No	Bus, Branch, Solution Values
<code>CHANGES</code>	ASCII	Input or Output	User <code>gui</code>	Yes	Bus, Branch, Modifications
<code>NEW_BASE .bse</code>	Binary	Output Only	IPF	No	Bus, Branch, Solution Values
Printout le ( <code>&lt;name&gt;.pfo</code> )	ASCII	Output Only	<code>bpf</code>	No	Input Data and Solution Reports, User Analysis
Microche le ( <code>&lt;name&gt;.pff</code> )	ASCII	Output Only	<code>bpf</code>	No	Input Data and Solution Reports, User Analysis
Debug le ( <code>&lt;name&gt;.pfd</code> )	ASCII	Output Only	<code>bpf</code>	No	Solution arrays and iteration processing
Printout le ( <code>&lt;logon&gt;.pfo</code> )	ASCII	Output Only	<code>gui</code>	No	Messages, Iteration Summary
Debug le ( <code>&lt;logon&gt;.pfd</code> )	ASCII	Output Only	<code>gui</code>	No	Solution arrays and iteration processing



## 2.2.4 The BASE (.bse) File

This file, designated OLD\_BASE if you are loading it, or NEW\_BASE if you are saving it, is binary in format and contains the following data:

- The case identification, project ID, and two header records.
- The date the case was generated.
- The program version used to generate the file (so future program versions can read the file if file structures change).
- Up to 100 comment records.

## 2.3 History

IPF was developed by [Bonneville Power Administration](#) (BPA) and its contractors in the 1990s with about 20% of the cost supported by the [Electric Power Research Institute](#) (EPRI). By mutual agreement, as described in EPRI Agreement RP2746-03 entitled *Graphical User Interface for Powerflow*, March, 1992,

“all results of this project—including the computer program and its documentation—are to be in the public domain.”

In a separate *Memorandum of Understanding with the Western Systems Coordinating Council* (WSCC), BPA agreed in March, 1992, to

“keep WSCC informed of progress, to make its best effort to develop the program according to the Guidelines adopted by the WSCC Computer Program Management Subcommittee, and to make the final results available for possible further development by WSCC.”

Prior to the development of the Interactive Power Flow in the 1990s, BPA had developed the core power flow program and a set of tools they called the Power System Analysis Package (PSAP), which were a collection of Fortran-coded computer programs permitting the analysis of the steady-state operation of an electric power network. PSAP was developed over a 20-year period and many features were added during this time. Since the core power flow code was developed in 70s and 80s, it had to make efficient use of computer memory space and computation time, the core engine uses traditional techniques of large power system analysis including the Newton-Raphson method of solution of algebraic equations and sparse matrix computation techniques.

The Western Systems Coordinating Council (WSCC) now known as the [Western Electricity Coordinating Council](#) (WECC) had a similar set of tools known as Interactive Powerflow System (IPS) complete with a Domain Specific Language called Computationally Oriented Programming Environment (COPE) for defining power flow studies in a programmatic way. Many of the features and data record formats for IPF, especially for transient stability, were influenced by WSCC’s IPS.

These programs were designed for character-based terminals connected to mainframes. However, the 1980s and 90s saw the creation and maturation of graphical user interfaces (GUI) based on the mouse, windows, and menus. These components were generally available on the Macintosh computer since 1984 and on IBM PC-compatible computers as an option since 1987. The third most popular GUI in the 90s was the one associated with the X Window System. GUIs were more intuitive and easier to use than character/keyboard-based interfaces. Starting in 1991, BPA began a two-way joint development of a GUI interface for the powerflow program, based upon the X-Window System and the [Motif X Window](#) GUI. The joint partners were the [Electric Power Research Institute](#) EPRI and WSCC. The powerflow program itself was also restructured and enhanced in the process, and a `libcflow` library was added as a means to provide users API access to the IPF functions for use in other programs.

The [Bonneville Power Administration](#) (BPA) Transient Stability Program was developed in accordance with specifications written by the Western Systems Coordinating Council (WSCC), now known as the [Western Electricity Coordinating Council](#) (WECC), and was originally written to be executed on a VAX 11/780 computer. With some exceptions this program closely paralleled the program used in Salt Lake City by the WSCC Technical Staff in the 1980s.

A copy of the original WSCC specifications for both power flow and transient stability can be found in *WSCC Specifications*.

The difference between this version of the program (i.e. `tsp`) and the one used in Salt Lake City was the control record requirements for defining the problem such as where the fault is applied, line switching, etc., and the format for all output options. The formats for almost all other input data requirements maintain strict compatibility between the two programs. Therefore, with grateful acknowledgements to the WSCC Staff, a substantial portion of the materials used in this documentation are reproduced copies taken from the WSCC Program Manual.

Acknowledgement is given to Teshmont Consultants, Inc. of Winnipeg, Manitoba for their many corrections and additions to the transient stability code.

### 2.3.1 Original License

Below is a copy of the original license notice from BPA.

The BPA Powerflow software (IPF) **is** Public Domain; here **is** a copy of the License.

#### IMPORTANT NOTE:

IPF **is in** the public domain **and** certain conditions apply. The terms **in** which it **is** released are described **in** the section "Notice to Non-BPA Users" **in** each of the IPF manuals.

#### NOTICE TO NON-BPA USERS:

The Bonneville Power Administration (BPA) releases BPA-developed computer programs under the following conditions:

1. BPA does **not** charge **for** program development costs; however, a fee to cover costs incurred **in** answering inquiries **is** assessed against the organization receiving the material. This fee typically includes costs **for** personnel **and** computer resources, reproduction, shipping, **and** postage.
2. BPA cannot provide assistance **with** conversion to other computers **or** consulting services to the program users.
3. In consideration of receipt **and** acceptance of these programs **or** portions thereof, **if** sold, assigned, **or** transferred to another organization, you **and** your organization agree to advise **any** third-party recipient **in** writing that the program(s) **and/or** documentation are **in** the public domain **and** available **from BPA**. The intent of this agreement **is** to ensure that BPA-developed **or** supplied programs, **and/or** documentation, whether **in** whole **or in** part, that are **in** the public domain, are identified **as** such to recipients.

#### "LEGAL NOTICE"

Neither BPA nor **any** person acting on behalf of BPA:

1. Makes **any** warranty **or** representation, expressed **or** implied, **with** respect to the accuracy, completeness, **or** usefulness of the information contained **in** this report, **or** that the use of **any** information, apparatus, method, **or** process disclosed **in** this report may **not** infringe upon privately owned rights; **or**

(continues on next page)

(continued from previous page)

2. Assumes any liability with respect to the use of, or for damages resulting from the use of any information, apparatus, method or process disclosed in this report.

9/20/95

In keeping with the license, please know that original code is in public domain. Also, you should be able to obtain the original code released by BPA by contacting [Bonneville Power Administration](#). However, for your convenience the v0.0.1 release of this project contains the original code released by BPA.

### 2.3.2 Original Contributors

The original project is the result of a lot of hard work from engineers at BPA and EPRI. These include, but are not limited to:

- Dan L. Clark
- Jay G. Coleman
- Tsu-huei Liu
- [Walter L. Powell](#)
- Bill E. Rogers
- William D. Rogers
- K E Rowell
- John L. Rutis
- David M. Stefonik
- D M Syzmanski

## 2.4 Application Examples

### 2.4.1 Introduction

Some typical power system design application examples are given in this section to demonstrate possible applications of the core power flow programs IPF provides. Each example indicates possible uses of the program relative to some stated network planning objectives.

### 2.4.2 Setting Up a Network Data File

Use the following values for this four-bus network:

#### Line Data

- Use 100 MVA, 500 kV base.
- $X = .525$  ohm/mile,  $X/R = 18$ .
- $B/2 = .01024$  pu/mile, where  $B$  is the total line charging and  $B/2$  is equal to  $B_{pi}$ , if long-line effect is ignored.

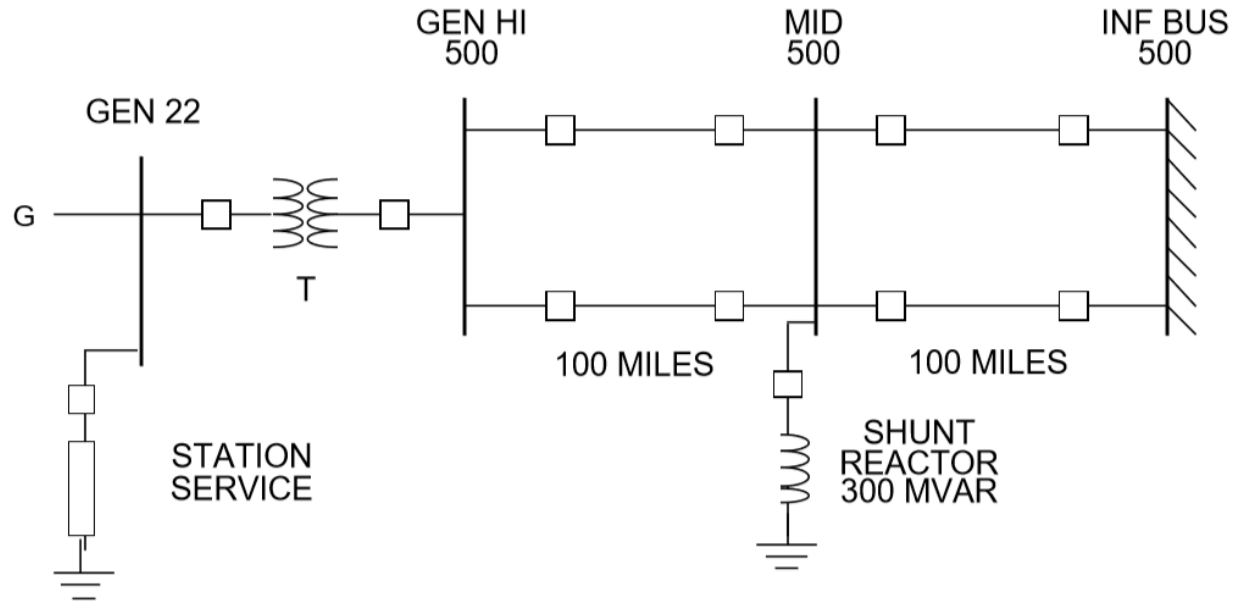


Fig. 2.4.1: Four Bus Network

- Current Rating is 3000 amps.

#### Transformer Data

- $X = 20\%$  (0.20 pu) on 525 kV, 1770 MVA base.
- Transformer Taps = 22 kV, 525 kV.

#### Generator Data for Power Flow

- $P(\text{max}) = 1770\text{MW}$ ,  $P(\text{g}) = 1680\text{ MW}$ .
- $Q(\text{lim}) = \pm 521\text{MVAR}$ .
- Type G bus controlling voltage of GEN HI to 1.08 pu.

#### Station Service Load

- $190 + j95$  (constant power for stability).

**Approach** The preparation of the network data will be presented in the following steps:

1. Prepare the bus data group.

Define the following buses:

```
GEN 22
GEN HI 500
INF BUS 500
MID 500
```

- a. Data for GEN 22

```
Record Type: B
Sub Type: G
Change Code: blank
```

(continues on next page)

(continued from previous page)

```

Owner:          blank (not specified)
Name:           GEN
kV:             22
Zone:           blank (not specified)
Load P MW:      190
Load Q MVAR:     95
Shunt Load MW:  blank (or zero)
Shunt Load MVAR: blank (or zero)
P MAX:          1770
P GEN MW:        1680
Q MAX MVAR:      521
Q MIN MVAR:      -521
V MAX PU:        blank (not specified)
V MIN PU:        blank (not specified)
Remote Bus Name: GEN HI
Remote Bus kV:   500
% VARS Supplied: blank (defaults to 100)

```

## b. Data for GEN HI 500

```

Record Type:    B
Sub Type:       C (remotely controlled)
Change Code:    blank
Owner:          blank
Name:           GEN HI
kV:             500
Zone:           blank
Load P MW:      blank (or zero)
Load Q MVAR:     blank (or zero)
Shunt Load MW:  blank
Shunt Load MVAR: blank
P MAX:          blank
P GEN MW:        blank (or zero)
Q MAX MVAR:      blank
Q MIN MVAR:      blank
V Hold PU:       1.08
V MIN PU:        blank (not used)
Remote Bus Name: blank (none)
Remote Bus kV:   blank (none)
% VARS Supplied: blank (not applicable)

```

## c. Data for INF BUS 500

```

Record Type:    B
Subtype:        S
Change Code:    blank
Owner:          blank
Name:           INF BUS
kV:             500
Zone:           blank
Load P MW:      blank
Load Q MVAR:     blank

```

(continues on next page)

(continued from previous page)

```

Shunt Load MW:   blank
Shunt Load MVAR: blank
P MAX:           blank
P GEN MW:        blank
Q SCHED MVAR:    blank
Q MIN MVAR:      blank
V Hold PU:       1.05
V MIN PU:        blank
Remote Bus Name: blank
Remote Bus kV:   blank
% VARS Supplied: blank

```

## d. Data for MID 500

```

Record Type:      B
Sub Type:         blank
Change Code:      blank
Owner:           blank (not known)
Name:            MID
kV:              500
Zone:            blank
Load P MW:       blank (zero)
Load Q MVAR:     blank (zero)
Shunt Load MW:   blank (zero)
Shunt React MVAR: -300
P MAX:          blank
P GEN MW:       blank
Q MAX MVAR:     blank (fixed)
Q MIN MVAR:     blank (fixed = Q MAX)
MAX PU:         blank (defaults to global limit)
V MIN PU:       blank (defaults to global limit)
Remote Bus Name: blank
Remote Bus kV:  blank
% VARS Supplied: blank (not applicable)

```

**Note:** A bus can only be of one subtype. The subtype of a bus is suggested by information given about the bus. Therefore, the user should try to be familiar with various bus subtypes and when and how they are indicated by descriptive information given

## 2. Prepare the branch data group.

```

Transformer:  GEN 22          GEN HI 500
Line:        GEN HI 500      MID 500 circuit 1
Line:        GEN HI 500      MID 500 circuit 2
Line:        INF BUS 500     MID 500 circuit 1
Line:        INF BUS 500     MID 500 circuit 2

```

**Note:** 1 and 2 designations are arbitrary identifications for parallel branches. Letters A-Z and digits 0-9 are acceptable.

## a. Data for Transformer GEN 22 GEN HI 500

```

Record Type:      T
Sub Type:         blank
Change Code:      blank
Owner:            blank (not known)
Name 1:           GEN
kV 1:             22
Meter:            blank
Name 2:           GEN HI
kV 2:             500
ID:               blank
Section:          blank
Total MVA RATE:   1770
No of CKT:        blank
Z_pi
Rpu(100MVA):      blank (or zero)
Xpu(100MVA):      .01246
Y_pi
Gpu(100MVA):      blank (or zero)
Bpu(100MVA):      blank (or zero)
Tap 1 kV:         22
Tap 2 kV:         525
Date In:          blank (in)
Date Out:         blank (not out)

```

b. Data for Line GEN HI 500 MID 500 circuit 1

```

Record Type:      L
Sub Type:         blank
Change Code:      blank
Owner:            blank (not known)
Name 1:           GEN HI
kV 1:             500
Name 2:           MID
kV 2:             500
ID:               1
Section:          blank
Total AMP RATING: 3000
No of CKT:        blank (means 1)
Z_pi
Rpu:              .00117
Xpu:              .02100
Y_pi
Gpu:              blank (zero)
Bpu:              1.024
Miles:            100
DESC DATA:       blank
Date In:          blank (in)
Date Out:         blank (not out)

```

c. Data for Line GEN HI 500 MID 500 circuit 2

```

Record Type:      L
Sub Type:         blank

```

(continues on next page)

(continued from previous page)

```

Change Code:      blank
Owner:            blank (not known)
Name 1:           GEN HI
kV 1:             500
Meter:            blank
Name 2:           MID
kV 2:             500
ID:               2
Section:          blank
Total AMP RATING: 3000
No of CKT:        blank (means 1)
Z_pi
Rpu:              .00117
Xpu:              .02100
Y_pi
Gpu:              blank
Bpu:              1.024
Miles:            100
DESC DATA:       blank
Date In:          blank (in)
Date Out:         blank (not out)

```

## d. Data for Line INF BUS 500 MID 500 circuit 1

```

Record Type:      L
Sub Type:         blank
Change Code:      blank
Owner:            blank (not known)
Name 1:           INF BUS
kV 1:             500
Meter:            blank
Name 2:           MID
kV 2:             500
ID:               1
Section:          blank
Total AMP RATING: 3000
No of CKT:        blank (means 1)
Z_pi
Rpu:              .00117
Xpu:              .02100
Y_pi
Gpu:              blank (zero)
Bpu:              1.024
Miles:            100
DESC DATA:       blank
Date In:          blank (in)
Date Out:         blank (not out)

```

## e. Data for Line INF BUS 500 MID 500 circuit 2

```

Record Type:      L
Sub Type:         blank

```

(continues on next page)



(continued from previous page)

Change Code:	blank
Owner:	blank ( <b>not</b> known)
Name 1:	INF BUS
kV 1:	500
Meter:	blank
Name 2:	MID
kV 2:	500
ID:	2
Section:	blank
Total AMP RATING:	3000
No of CKT:	blank (means 1)
Z_pi	
Rpu:	.00117
Xpu:	.02100
Y_pi	
Gpu:	blank (zero)
Bpu:	1.024
Miles:	100
DESC DATA:	blank
Date In:	blank ( <b>in</b> )
Date Out:	blank ( <b>not</b> out)

**Note:** The most difficult task in setting up branch data is determining the per unit (pu) values of R, X, G and B. See [Calculating Line Impedance](#) for some help on this.

**Note:** In this example, some preliminary information has been furnished on a per-mile basis. We have multiplied the per-mile quantities by the line lengths in order to obtain the data entered in the record. The reader should generally use detailed line constant calculation procedures to obtain more accurate values for high-voltage long lines. At 500 kV, 100 miles should be considered a long line. Energy conservation analysis is sensitive to the accuracy of high-voltage long-line branch data and transformer branch data.

Derivation of Transformer Xpu on 500kV, 100 MVA:

$$\begin{aligned}
 (X_{pu \text{ on } 500kV 100MVA}) &= (X_{pu \text{ on } 525kV 1770MVA}) * \frac{525 * 525}{500 * 500} * \frac{100}{1770} \\
 &= 0.20 * \frac{525 * 525}{500 * 500} * \frac{100}{1770} \\
 &= 0.012458
 \end{aligned}$$

Derivation of Line Xpu on 500 kV, 100 MVA:

All four lines are the same:

$$\begin{aligned}
 \text{Base impedance in ohms} &= \frac{\text{Base kV} * \text{Base kV}}{\text{Base MVA}} \\
 &= \frac{(500)(500)}{100} \\
 &= 2500 \text{ ohms}
 \end{aligned}$$

Ignoring long-line effect, reactance for 100-mile line is 52.2 ohms:

$$\begin{aligned} &= \frac{52.5}{2500} pu \\ &= 0.02100 pu \end{aligned}$$

Derivation of Line Rpu on 500 kV, 100MVA:

All four lines are the same:

### 2.4.3 New Facilities

The purpose of this example is to provide new generating facilities to serve the growing loads near the town of Keller and Mount Tolman.

#### Proposed New Facility

The proposed facility includes a new substation at NEW SUB, a 21-mile 230 kV line tapped from the A SUB - B SUB No. 2 line and another line to D SUB tapped from A SUB - NEW SUB line but which will be built and owned by another utility. The equipment at NEW SUB substation will include a 25 MVA transformer with protective equipment serving a 34.5 kV load. In the diagram on the following page, starred or dotted lines are the proposed facility while the dashed lines indicate existing facility. This proposed facility may be one among other possible alternatives. Each alternative, and modifications thereof, will be considered a case and submitted to the power flow program for analysis and report.

#### Possible Uses of IPF

IPF will be used to study the power flows, voltage regulation and reliability of service utilizing the proposed facility.

The processes (POWERFLOW) and (OUTAGE\_SIM) will be used to check power flow as well as reliability. Refer to *Power Flow Control (PFC)*, for complete description of these processes.

Suppose the base network which the proposed new facility will amend has been described in an old base file named BASENET.BSE. The following program control file can be built for this case:

```
(POWERFLOW, CASEID = EXAMPLE1, PROJECT=KELLER-SUB)
/OLDBASE FILE = BASENET.BSE\
/NEWBASE = EXAMPLE1.CAS\
....
....  Optional Network Solution Qualifiers Listed Under
....                                     (POWERFLOW)
/CHANGES
B - record for NEW SUB 34.5
B - record for NEW SUB 230
B - record for NEW SUB Tap 230
B - record for D SUB 230
B - record for A SUB TAP 230
L - record to delete A SUB 230 B SUB 230 2
L - record for A SUB 230 A SUB Tap 230
L - record for A SUB Tap 230 B SUB 230
T - record for NEW SUB 230 NEW SUB 34.5
. The above record formats are described
. Under appropriate Heading starting with the key letter
```

(continues on next page)

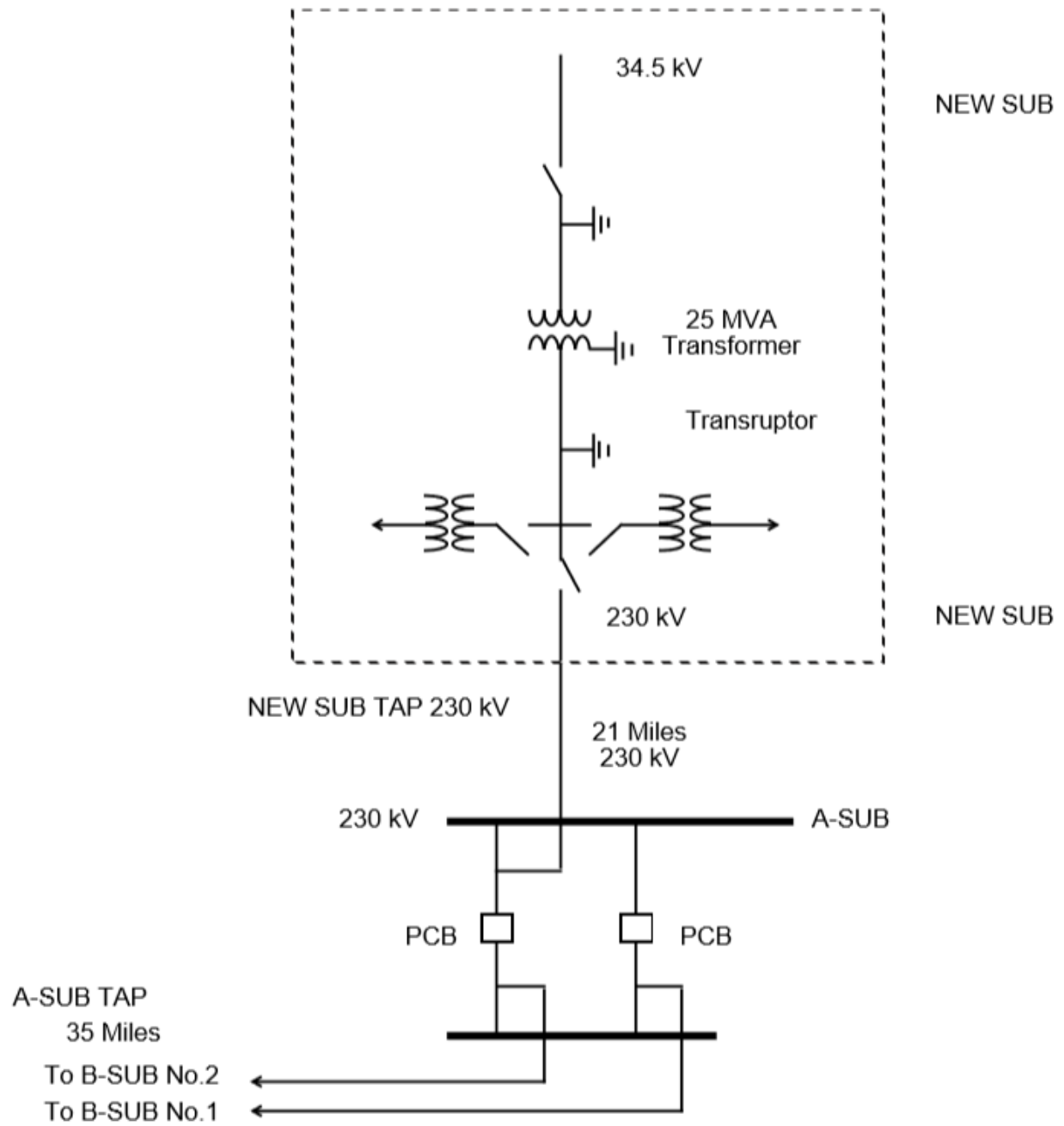


Fig. 2.4.2: A-SUB - New SUB 230 kV Line

(continued from previous page)

```
(POWERFLOW, CASEID=EXAMPLE1, PROJECT=NEW-SUB)
/OLDBASE FILE=EXAMPLE1.CAS
/OUTAGE_SIM
....
.... Optional Qualifiers Listed Under /OUTAGE_SIM
....
(STOP)
```

The output listings from this case will be inspected to make sure no overloads occur during normal operation as well as during the outages of key lines.

Special environmental concerns may suggest evaluation of alternatives to the river crossing indicated in the exhibit. Conductor sizing may also be influenced by energy conservation (loss-reduction) considerations. The effect of each alternative can be determined from the network solution output listing.

In this example, the major point to monitor is the impact of tapping lines since segments of a tapped line perform differently from the untapped line.

### 2.4.4 Reconductoring

The purpose of this example is to cure poor voltage regulation at delivery point C occurring when lines AB, BC or DE are out of service, and to improve energy conservation.

#### Proposed New Facility

Lines AB, BC and DE will be reconducted to reduce series impedances using higher capacity lines. The higher capacity lines will give rise to acceptable voltages and save energy otherwise lost in transmission.

#### Possible Uses of Powerow

IPF can be used to study the power flows, voltage regulation and reliability of service utilizing the proposed facility. In reference to the Program Control Language:

1. The base case is run to determine power flows, voltage levels and transmission losses.
2. The change case is run to modify the data for lines AB, BC and DE reflecting the new conductors used. Power flows, voltages and losses will again be reviewed.
3. The outage simulations case is run to verify the effect of certain lines being out of service on power flows, voltage levels, line loading and line losses. If the base network to be amended by the proposed new facility is described in an old-base file named BASENET.BSE, the program control file following the diagram of the reconducting can be built for this project.:

```
( POWERFLOW, CASEID = EXAMPLE30, PROJECT = RECONDUCT )
/ OLD_BASE, FILE = BASENET.BSE
. . .
. . . Optional network solution qualifiers.
. . .
( NEXTCASE, CASEID = EXAMPLE31, PROJECT = RECONDUCT )
/ NEW_BASE, FILE = EXAMPLE31.CAS
. . . Optional qualifiers to override options
. . . already selected above. This should be
```

(continues on next page)

(continued from previous page)

```

. . . an empty set.
/ CHANGES
L - record to modify line AB
L - record to modify line BC
L - record to modify line DE
( NEXTCASE, CASEID = EXAMPLE32, PROJECT = RECONDUCT )
/ OUTAGE_SIM
. . .
. . . Optional simulation qualifiers.
. . .
( STOP )

```

## 2.4.5 Series Compensation

The purpose of this example is to add series compensation to existing parallel lines so as to cause more power to be shifted to these lines from lines with less loss-reduction.

### Proposed New Facility

A 540 MVAR series capacitor at Station C is installed (270 MVAR per line), along with additional control and protective equipment. Station C is sited 72 miles from Station A and 102 miles from Station B.

### Possible Uses of Powerow

IPF will be used to study the power flows utilizing the proposed facility.

If the base network amended by the proposed new facility is described in an old-base file named `BASENET.BSE`, then the following program control file can be built for this case:

```

( POWERFLOW, CASEID = EXAMPLE4, PROJECT = SERIES COMP )
/ OLD_BASE, FILE = BASENET.BSE
. . .
. . . Optional solution qualifiers.
. . .
/ CHANGES
L - record to delete line AB circuit 1
L - record to delete line AB circuit 2
L - record to add Ckt circuit 1 section 1 (line AC)
L - record to add Ckt circuit 1 section 2 (capacitor)
L - record to add Ckt circuit 1 section 3 (line CB)
L - record to add Ckt circuit 2 section 1 (line AC)
L - record to add Ckt circuit 2 section 2 (capacitor)
L - record to add Ckt circuit 2 section 3 (line CB)
( STOP )

```

## 2.5 Record Formats

### 2.5.1 Overview

This section describes all data record formats recognized by IPF. Records are used to represent buses, lines, transformers, reactive devices, area interchange control, and more. Records defined in the input files for the command line tools (`bpf`, `ipfbat`, `ipfcut`, etc.) are fixed column format. Please note that all information must be in the correct columns; you will experience processing errors otherwise. Most records are also available in the X Window GUI interface via forms and in `libcflow` via C structs. The record descriptions in this section are organized alphabetically by record ID as you go through the section (except for the DC bus records which follow the AC bus records). The table below enables you to quickly locate a specific record format entry. The table also gives you a quick description of each record format.

Each record description in this section has a “card” figure that shows you where to put record identification and the other fields and properties of the record. These “card” figures are artifacts from the days of punchcards and mainframes, but they do provide good visual representation of the data for each record format when manually creating input files for IPF. Each card figure has a legend showing whether each field is required, optional, or ignored.

Each entry also has a table of column descriptions and data formats. This supplements the card figure. Additional text accompanies any record formats that need extended discussion.

Some data are real numbers. For these fields, the implicit decimal point is shown as a black dot on the card figure, just above the information entry row. Note that you can enter data with an explicit decimal point in any position within the field. A black dot that appears at the extreme right side of a field represents a decimal point at the right end (least significant digit) of a number. All other black dots appear above a line, indicating that the decimal point will fall between the digits on either side.

All IPF data record types are identified by the characters in columns 1 and 2. To fully specify a particular record, additional fields need to be filled in. These additional ID fields are usually the bus name and base kV. ID fields are called out in the column description tables.

Table 2.5.1: Record Types

Record ID (columns 1-2)	Field Width	Description
Period (.)	120	Comment (not printed)
+	80	Bus continuation
A	80	Area interchange control
A0	80	Area output sort
-	-	General AC bus description
B	80	AC bus - load bus
BC	80	AC bus - voltage controlled by BG bus
BE	80	AC bus - constant voltage
BF	80	AC bus - special purpose bus for Newton-Raphson solution
BG	80	AC bus - generator
BQ	80	AC bus - constant voltage within Q limits
BS	80	AC bus - system slack bus
BT	80	AC bus - LTC transformer controlled AC bus
BV	80	AC bus - constant Q within V limits
BX	80	AC bus - attempts constant V using switched Q specified on X record
BD	80	Two-terminal DC bus
BM	80	Multi-terminal DC bus
DA	80	Delete buses by area
DZ	80	Delete buses by zones
E	88	Equivalent branch (has extended ratings)

continues on next page

Table 2.5.1 – continued from previous page

Record ID (columns 1-2)	Field Width	Description
I	80	Area intertie I record
L	88	Transmission line (has extended ratings)
LD	80	Two-terminal DC line
LM	80	Multi-terminal DC line
PO PZ PN PA PB PC PD	80	Factor changes
QN QP QX	120	Reactive capability curve
R RV RQ RP RN RM	80	Regulating transformer
RZ	80	VAR compensator model
T	92	Transformer (has extended ratings)
TP	92	Phase shifter (has extended ratings)
X	80	Switched reactance (BX record)
Z	80	Zone rename

## 2.5.2 System Changes

After a base case has been established, it may be changed with the use of change records. The change records are identified as system data records which immediately follow the network-solution qualifier /CHANGES statement. See [CHANGES](#) for details.

The change records are system data records with a change code in column 3. Each of the input data forms describe the permissible types of changes under the caption CHANGE CODES. In all, there are five types of changes.

- *Additions:* Change code = blank

The data record identification must be unique to the system. The contents of the record must be complete as if it were being submitted to build a base case.

- *Deletions:* Change code = D

Only existing data may be deleted, and only the identification fields are needed. Numerical data in any other field is ignored. Special conditions are given:

- Deleting a bus automatically deletes all continuation bus data, switched reactance data and all branch data associated with that bus. Deleting all branches connected to a bus will result in an error. It is better to delete the bus. A bus should not be deleted and added back in the same case with the same name in a single change file.
- Deleting all transformer banks between two buses will automatically delete any regulating transformer data.
- A line composed of sections may be deleted in its entirety by deleting section 0 (zero) or blank. The alternative is to delete each section with a separate change record. If a section is deleted the line is reconnected without that section. Transformer sections cannot be deleted.
- A branch composed of parallel lines between two buses may be deleted in its entirety by entering a \* in place of CKT ID. This provides a means of disconnecting two directly connected buses from each other. The branches may also be deleted individually.
- A blank branch ID is legitimate identification.
- Changing a bus from type X to any other type will delete all switched-reactance data automatically. No separate X delete record should be included.

- *Elimination:* Change code = E

This causes the elimination of all existing A (area interchange) records to make room for possible new A records. The user should note that this change code works for A records only.

- *Modifications*: Change code = M

All currently existing system data may be modified, which means changing the value of some quantity of system data. Data in the identification field cannot be changed.

Only the data to be changed is entered on the change record. Nonblank fields constitute data modification, while blank fields indicate that the quantity is not to be changed. A blank and a zero quantity on the change record are distinguishable. Often, it is necessary to change a quantity into a blank. Examples are the bus zone name and the bus subtype. To change these into blank quantities, a 00 or 0 must be entered in the appropriate respective column fields. Special recognition is conferred on these change quantities.

- *Restorations*: Change code = R

Previously outaged system data may be reactivated with the use of restore change records. This type change permits data to be restored to the system with change records using the identification fields only. Any data fields on the restore record will be ignored, and the reactivated data assumes the same quantities it had prior to deletion. Restoring data is an option. Data could be re-entered with additional type change records as well. The operations and rules are similar to those for deletions. However, some important differences must be explained.

- Restore changes are permissible only with data deleted in a previous change case but within the same base case. You cannot restore data outaged in a previous OLD\_BASE case.
- Area interchange records may not be restored.
- Restoring a bus restores only branches which connect to a viable system base. In some cases, not all of the outaged branches can be restored.
- Restoring all transformer banks between two buses will not automatically restore any regulating transformer data. If this is intended, the regulating R transformer data must be restored separately.
- A restored branch record must match the branch code as well as the identification fields. For example, if the branch is type L, the restore record must match the type.

The change records are read, interpreted and stored for further processing. As they are read the following are processed immediately:

- Area Interchange (A-blank)
- Zone Renames (Z-blank)
- Area Deletes (DA)

Processing of changes then continues in the following manner:

- Changes Percents (P)
- Deletes
- Adds/Restores
- Modifications

---

**Note:** If more than one modification for the same data item occurs, the changes will be made in the order encountered. In IPF, no message will be given the user.

---

If fatal data errors are encountered in batch mode, switches are set and the processing continues only to discover additional errors, list all changes and exit. For changes to existing data, component identification must be specified exactly as in the base case file. If fatal errors are encountered in interactive mode, appropriate diagnostics are issued so that the user can remedy the faulty command or data.



### 2.5.3 Comment

This comment text is used to annotate the program control file, network data file, or change file. Its contents are not added to the output listings nor saved in the binary base case file.

Simply place a . (period character) in the first column and the comment in the rest of the columns. This is also handy to temporarily deactivate a command or data line.

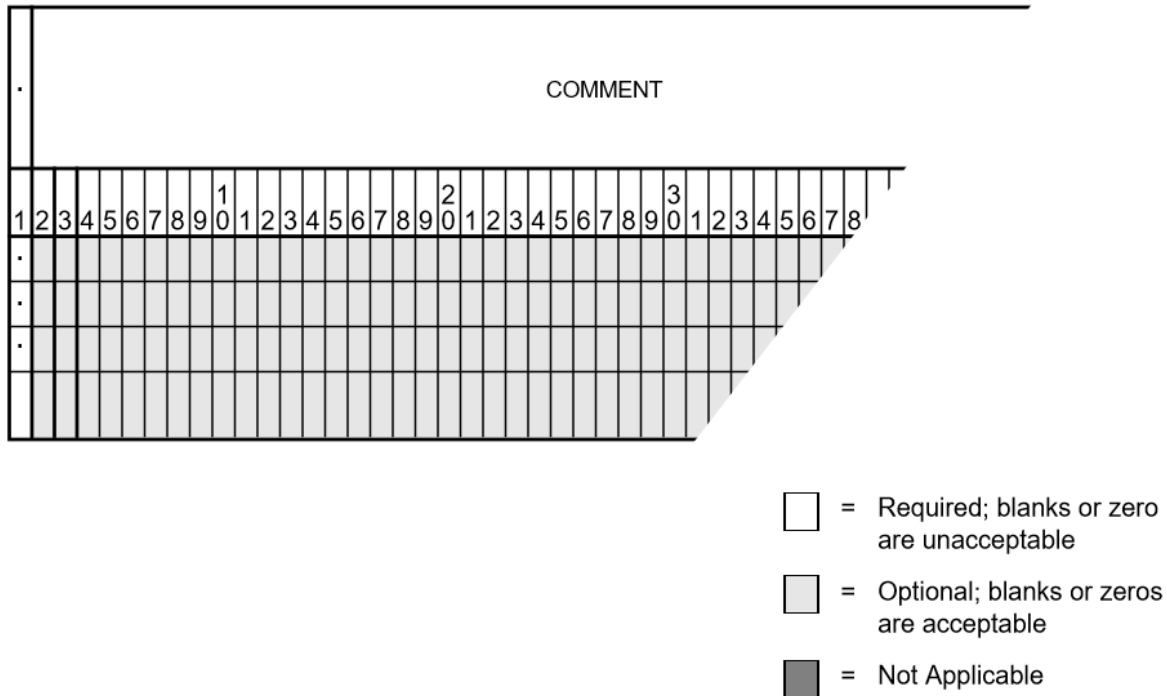


Fig. 2.5.1: Comment Input Format

Table 2.5.2: Column Descriptions for Comment Format

Column	ID Field	Format	Content
1	yes	A1	. (period)
2-120	no	A119	Text string for record comment

### 2.5.4 Continuation Bus Data (+)

Continuation bus data is identified with a + in column 1 and supplements the data on any AC bus record. It specifies additional generation, load and shunt admittance at the bus and permits additional classification and utilization of data. Generally, it permits a more detailed analysis of data. Its most typical application is distinguishing loads represented by several different owners at the same bus. The following fields are for identification:

- Bus name and base kV
- Code and code year
- Ownership

Each bus may have more than one continuation record. However, some means of distinction must be made in the minor identification fields of code year or owner. See figure and tables below for details.



Table 2.5.4: Special Constant Current and Constant Impedance Loads

Code	Code year	P_load	Q_load	G_shunt	B_shunt	Description
+A		Constant power MW load (generation if negative)	Constant power MVAR load (generation if negative)	Constant admittance MW evaluated at nominal voltage	Constant admittance MVAR evaluated at nominal voltage	Quantity generated by Network Data or Cutting routines
+A	00	Constant power MW load (generation if negative)	Constant power MVAR load (generation if negative)	Constant admittance MW evaluated at nominal voltage	Constant admittance MVAR evaluated at nominal voltage	Quantity generated by Network Data routine
+A	01	Distributed constant current MW load (generation if negative) evaluated at nominal voltage	Distributed constant current conjugate MVAR load (generation if negative) evaluated at nominal voltage	Equivalent MW shunt admittance	Equivalent MVAR shunt admittance	Quantity generated by Network Reduction routines
+A	02	Distributed MW load (generation if negative)	Distributed MVAR load (generation if negative)	Equivalent MW shunt admittance	Equivalent MVAR shunt admittance	Quantity generated by Network Reduction; denote equivalent shunt admittances
	*I	Constant current MW load (generation if negative) evaluated at nominal voltage	Constant current conjugate MVAR load (generation if negative) evaluated at nominal voltage	Not applicable	Not applicable	Quantity generated by %LOAD_DISTRIBUTION
	*Z	Constant power MW load (generation if negative)	Constant power MVAR load (generation if negative)	Constant admittance MW load (generation if negative) evaluated at nominal voltage	Constant admittance MVAR load (generation if negative) evaluated at nominal voltage	Quantity generated by %LOAD_DISTRIBUTION
	*P	Constant power MW load (generation if negative)	Constant power MVAR load (generation if negative)	Not applicable	Not applicable	Quantity generated by %LOAD_DISTRIBUTION

Table 2.5.5: Column Description for Continuation Bus Data

Column	ID Field	Format	Description
1	yes	A1	Record type; + for all continuation bus data
2	yes	A1	Code (See code types above.)
3	no	A1	Change code
4-6	yes	A3	Ownership
7-14	yes	A8	Bus name
15-18	yes	F4.0	Base kV
19-20	yes	A2	Code year—alphanumeric subtype of code
21-25a	no	F5.0	Load MW
26-30	no	F5.0	Load MVAR
31-34	no	F4.0	Shunt Admittance Load in MW at base kV
35-38	no	F4.0	Shunt Admittance in MVAR at base kV (+) = Capacitive (-) = Inductive
43-47	no	F5.0	P GEN MW
48-52	no	F5.0	Q GEN MVAR (or Q MAX) (+) = Capacitive (-) = Inductive
53-57	no	F5.0	Q MIN in MVAR
75-77	no	A1, A2	Energization date month and year {month = 1,2,3,4,5,6,7,8,9,O,N,D}

a. If the Code (column 2) is **A** and the Code year (column 19-20) is **01**, the load quantities are constant current-constant power factors and are:

1. Interpreted as MW and MVAR evaluated at base kV (+) = Inductive (-) = Capacitive
2. Evaluated as:  $P + jQ = (I^*)|V|$

**Note:** Q\_max and Q\_min represent inequality constraints ( $Q_{\max} > Q_{\min}$ ). However special concerns apply for bus subtypes blank, C, T and V, where the Q is constrained. In order to assign appropriate values for scheduled Q, Q\_min is first examined. If  $Q_{\min} < 0$  and  $Q_{\max} > 0$ , the limits are recognized as erroneous constraints and both are ignored. If  $Q_{\min} = 0$ , then the schedule Q is always Q\_max

## 2.5.5 Area Interchange Control (A)

A network may be partitioned geographically by area. Similarly, areas may be partitioned by zones. The net power exported from each area can be specified with an area interchange record. Export power is controlled by varying the area slack bus generation. Interchange export is measured as the sum of the exported power on all area tie lines metered at the area boundaries. The total net export of all areas must add to zero; otherwise, the area interchange control is aborted. See the figure and tables below.

Valid subtypes are blank, 1, ..., 9, with 1, ..., 9 being continuation records. This allows up to 100 zones to be defined in an area.

In order for area interchange control to be activated, A records must be defined and the /AI\_CONTROL option must be set to the default, CON (Control). Three slack bus restrictions pertain to each area.

- One area slack bus must be the system swing bus.
- Each area slack bus must be within the area it controls.
- For all slack buses the P generation is variable.

**Note:** Area Continuation records (A1, ..., A9) accept only area name and zones 1-10 fields.

SCH UH BG TC PD EE	INTERCHANGE AREA NAME										AREA SLACK BUS NAME		SCHEDULED EXPORT	ZONE										MAX P.U. VOLT	MIN P.U. VOLT					
											NAME	BASE kV		1	2	3	4	5	6	7	8	9	10							
A	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
A																														
A1																														
A2																														
A3																														
A4																														

NAME and BASE  
kV FIELD:  
Not required for  
changes.

SCHED. EXPORT FIELD:  
Not required for changes.

= Required in most cases; blanks or zeros  
are unacceptable; exceptions are noted.

= Optional; blanks or zeros  
are acceptable.

= Not Applicable.

Fig. 2.5.3: Area Interchange Control Input Format


Table 2.5.6: Column Description for Area Interchange


Col- umn	ID Field	For- mat	Description
1	yes	A1	Record type — A
2	yes	A1	Subtype — blank, 1, ..., 9
3	no	A1	Change code — see System Changes
4- 13	yes	A10	Interchange area name — Name of area consisting of one or more zones. Alphanumeric entries are permitted.
14- 25	no	A8,F4.0	Area slack bus name and base kV. (Does not apply to subtypes A1, ..., A9.)
27- 34	no	F8.0	Scheduled export — MW ow scheduled (+) out of area or (-) into area. If I (interchange) records are present for this area, the net schedule will be overwritten with the netting computed from the I records. (Does not apply to subtypes A1, ..., A9.)
36- 64	no	10(A2,Z8)	Zones to be included in the interchange area named in columns 4-13. A blank zone terminates the scan unless it is zone 1. All zones must be listed within some area, but no zone may be common to more than one area.
73- 76	no	F4.3	Maximum per unit voltage. (Does not apply to subtypes A1, ..., A9.)
77- 80	no	F4.3	Minimum per unit voltage. (Does not apply to subtypes A1, ..., A9.)


## 2.5.6 Area Output Sort (A0)

The order of buses in the input and output listings may be grouped into areas with arbitrary zone configurations. The areas are sorted alphabetically and the buses within each area are then sorted alphabetically.

These records permit an area to be defined independently of any area interchange. Once introduced, these records become a permanent part of the base case file. A coding sheet and description of the record columns follows:

 = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.

 = Optional; blanks or zeros are acceptable.

 = Not Applicable.

Column	ID Field	Format	Description
1	yes	A1	Record type — A
2	yes	A1	Subtype — O
3	no	A1	Change code — see System Changes
4-13	yes	A10	Area Name — These names are independent of area interchange names but may be identical.
15-79	no	22(A2,1X)	Zone composition list — a blank zone terminates the zone scan unless it is zone 1.
80	no	blank	

Column	ID Field	Format	Description
1	yes	A1	AC B type record
2	no	A1	Subtype
3	no	A1	Change code
4-6	no	A3	Ownership
7-14	yes	A8	Bus name
15-18	yes	F4.0	Base kV
19-20	no	A2	Zone
21-26	no	F5.0	Load MW
26-30	no	F5.0	Load MVAR
31-34	no	F4.0	Shunt Admittance Load in MW at base kV
35-38	no	F4.0	Shunt Admittance in MVAR at base kV (+) = Capacitive (-) = Inductive
43-47	no	F5.0	P GEN MW
48-52	no	F5.0	Q GEN MVAR (+) = Capacitive (-) = Inductive
53-57	no	F5.0	Q MIN MVAR
58-61	no	F4.3	V HOLD - V MAX (in per unit)
62-65	no	F4.3	V MIN (in per unit)
66-73	no	A8	Controlled bus name
74-77	no	F4.0	Base kV
78-80	no	F3.0	Percent of vars supplied for remote bus voltage control.

Allocation of reactive facilities is complex. These may be allocated by equality constraints ( $Q_{net}$  is constant), inequality constraints ( $Q_{net}$  varies between a minimum and maximum value), or no constraints.

$$Net = Generation - Load - Y_{shunt} * Voltage^2$$

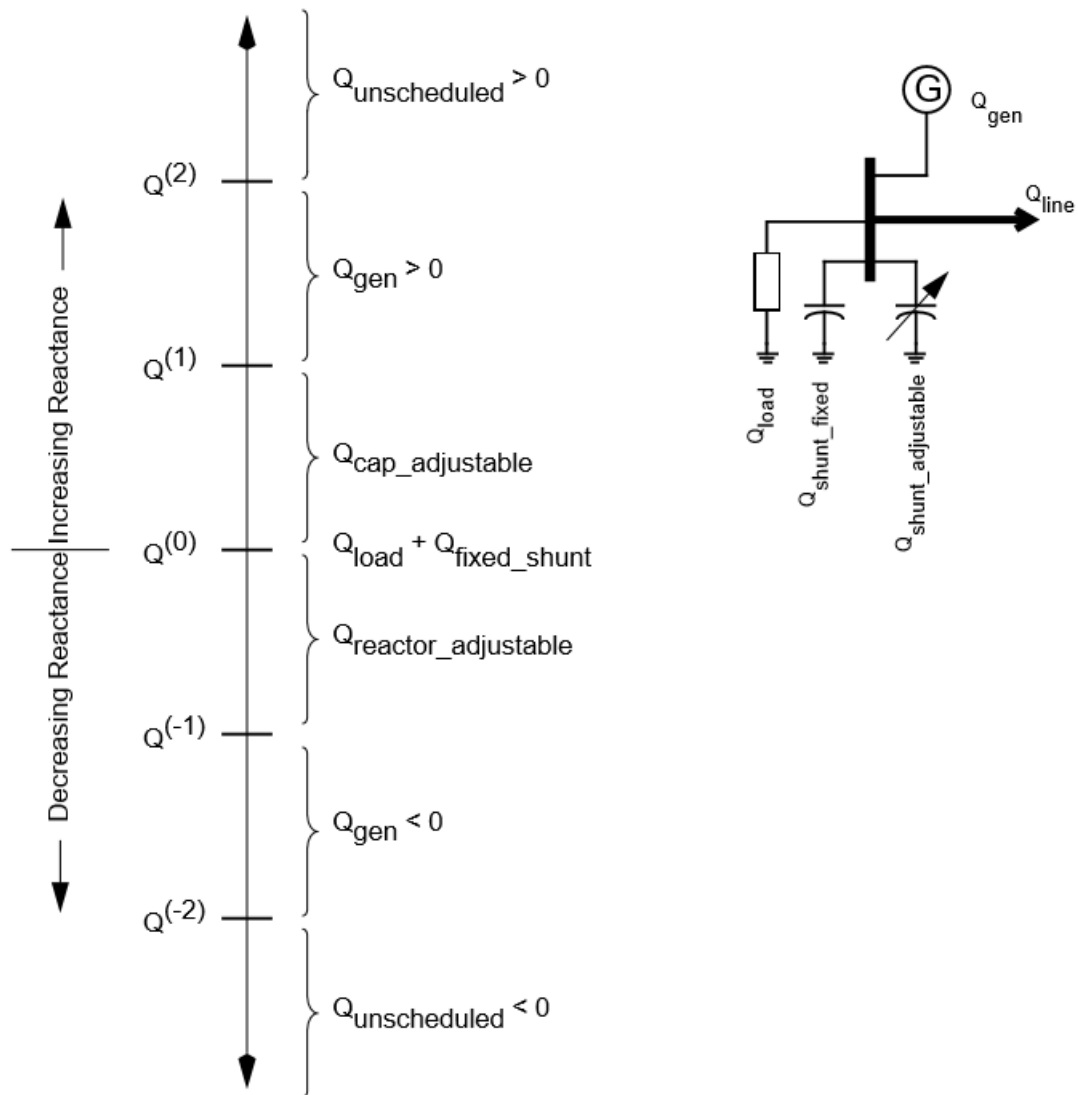


Fig. 2.5.6: Reactive Allocation Scheme



The equation is complex; the real and reactive components are balanced separately. The separate equations are:

$$P_{net} = P_{Gen} - P_{load} - G * V^2$$

$$Q_{net} = Q_{Gen} - Q_{load} + B * V^2$$

The shunt admittance is

$$Y = \frac{1}{Z} = G + jB$$

The equations above define the interrelationship between quantities and are valid for all bus types. The bus type determines which equations are also constraints.

The preceding diagram illustrates the following priority scheme. In applying the equation for  $Q$ , vars are allocated to generation and variable shunt components on a priority basis. If  $Q_{net}$  is less than  $Q_{load} + Q_{shunt}$  fixed, then vars are allocated first to variable shunt reactors and then, if necessary, to reactive generation. If, on the other hand,  $Q_{net}$  is higher than  $Q_{load} + Q_{shunt}$  fixed, then vars are allocated first to the variable shunt capacitors and then, if necessary, to reactive generation. If the limits of reactive generation are exceeded, then unscheduled reactive is allocated.

## 2.5.8 AC Bus Data (B-blank)

### Application

This bus subtype is passive in the sense that it cannot control the voltage of another bus. Its primary use is for modeling load buses.

### Bus Characteristics

Both real ( $P$ ) and reactive ( $Q$ ) power are held constant throughout the entire solution. This applies to generators, load and shunt devices (capacitors/reactors).

A specific amount of reactive generation can be requested. This can be accomplished by entering a zero (0) in the Q MIN field and the desired amount of reactive generation in the Q SCHED field.

Since this bus normally has no voltage control, the voltage limits (V MAX, V MIN) serve two purposes.

- If the bus is remotely controlled by another bus (type BG or BX) or by an LTC transformer (which is not standard but is accepted), the limits specify the range of acceptable voltage.
- For accounting purposes, these limits can flag undervoltage or overvoltage situations in the analysis reports.

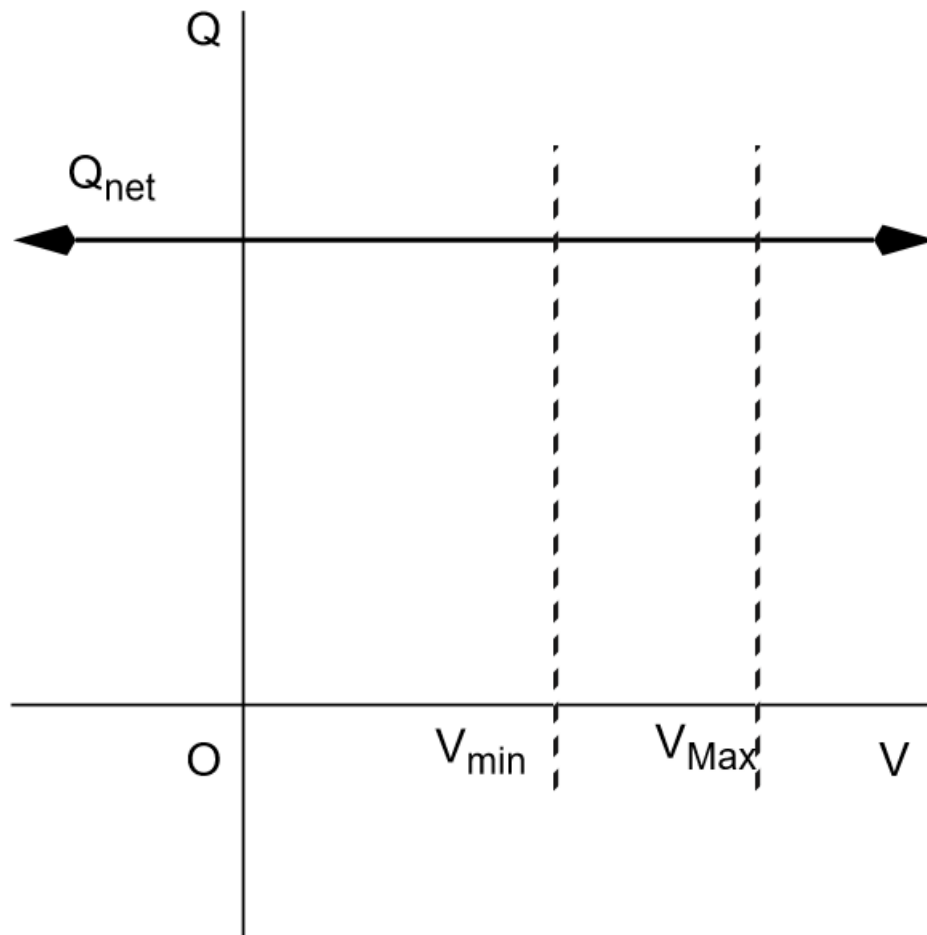
It must be recognized that every bus has voltage limits, whether they are explicitly specified through the V MIN, V MAX fields or implicitly specified through default global voltage limits. See [Voltage Limits and Starting Voltages](#) for details on global limits.

Reactive limits are not allowed for this type of bus. If reactive limits are entered in the Q MAX and Q MIN fields, they will be ignored. However, legitimate Q GEN can be entered if Q MIN is zero.

For this subtype,  $Q_{net}$  is constant; its Q-V characteristic is shown in format drawing below.

If this bus is controlled by an LTC transformer or by a BG or BX bus, a warning diagnostic will be issued to the effect that remotely controlled buses are typically type BC or type BT and the controlled voltage is a single value,  $V_{sched}$  and not a range  $V_{min} < V_{controlled} < V_{max}$ .

B	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



$Q_{\text{net}}$  is constant for all voltages.

Fig. 2.5.8: Q-V Curve for B-blank Subtype

### 2.5.9 AC Bus Data (BC)

## Application

This bus type has its voltage maintained by a subtype BG bus.

## Bus Characteristics

Both real ( $P$ ) and reactive ( $Q$ ) power are held constant throughout the entire solution. This applies to generators, loads, and shunt devices (capacitors/reactors).

A specific amount of reactive generation can be requested. This can be accomplished by entering a zero (0) in the Q MIN field and the desired amount of reactive generation in the Q SCHED field.

Reactive constraints are not allowed for this type of bus. If reactive limits are entered in the Q MAX and Q MIN fields, they will be ignored.

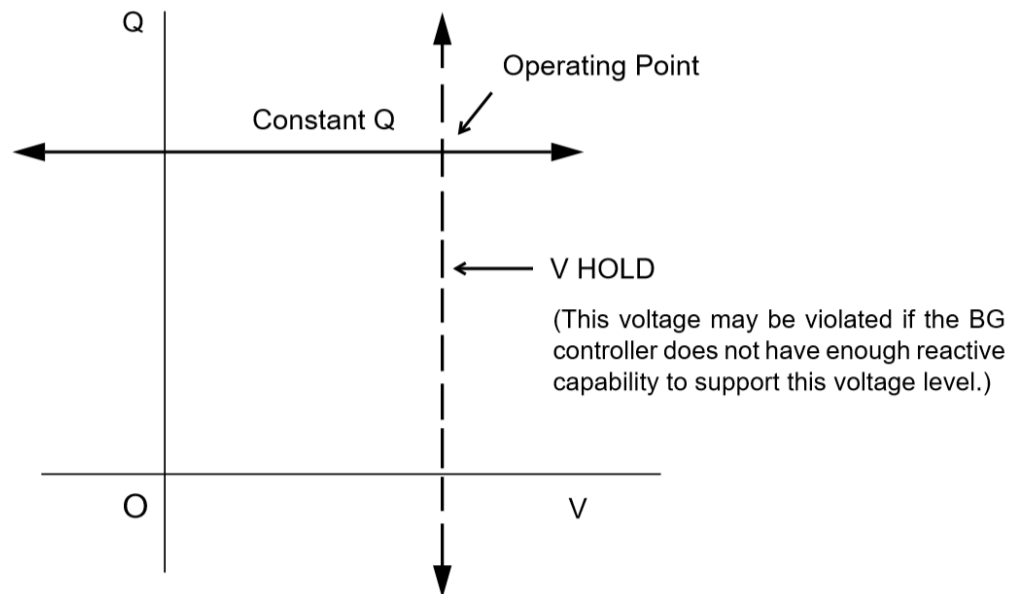
Since this bus type has its voltage maintained by a generator bus, a V HOLD entry is strongly recommended on the bus record. However, if that field is blank, the global default limits apply, in effect, using VMAX for the VHOLD. See [Voltage Limits and Starting Voltages](#) for details on global limits.

[illegible]

Fig. 2.5.9: BC Subtype Format

Table 2.5.10: Column Description for BC Bus Data

Column	ID Field	Format	Description
1-2	yes	A2	BC - Voltage controlled by BG bus
3	no	A1	Change code
4-6	no	A3	Ownership
7-18	yes		Bus name and base kV
15-18	yes	F4.0	Base kV
19-20	no	A2	Zone
21-25	no	F5.0	Load MW
26-30	no	F5.0	Load MVAR
31-34	no	F4.0	Shunt Admittance Load in MW at base kV
35-38	no	F4.0	Shunt Admittance in MVAR
39-42	no	F4.0	P MAX
43-47	no	F5.0	P GEN
48-52	no	F5.0	Q SCHED in MVAR
53-57	no	F5.0	Q MIN — Must be blank or zero for Q SCHED to apply
58-61	no	F4.3	V HOLD in per unit
62-65	no	F4.3	N/A
66-77	no	A8,F4.0	N/A
78-80	no	F3.0	N/A



Bus voltage and Q are held constant. Voltage is maintained by a controlling subtype BG bus and is held independently of Q.

Fig. 2.5.10: Q-V Curve for BC Subtype

### 2.5.10 AC Bus Data (BE)

## Application

This subtype is used to hold the bus voltage to a specified value, regardless of the amount of reactive required.

## Bus Characteristics

Voltage magnitude ( $V$ ) is held constant. Real ( $P$ ) power is held constant. This applies to generators, load, and shunt devices. Reactive ( $Q$ ) load is held constant for this bus type.

Reactive ( $Q$ ) shunt is variable. The amount of shunt reactance added by the program can vary from 0 to  $Q_{shunt}$ , depending upon the amount needed to maintain desired bus voltage.

Reactive ( $Q$ ) generation is variable.

Reactive constraints are allowed for this bus type. These quantities are entered in the Q MAX and Q MIN fields. If reactive constraints are imposed, “unscheduled reactive” may be added by the program to hold the bus voltage.

A specific amount of reactive generation ( $Q_{\text{SCHED}}$ ) cannot be requested.

Since this bus type maintains its own voltage, a V HOLD entry is required on the record. The voltage is held fixed at this value, regardless of the amount of reactive required. Note that V HOLD is not required for type “M” changes.

B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN	BO	BP	BQ	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA	CB	CC	CD	CE	CF	CG	CH	CI	CJ	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT	CU	CV	CW	CX	CY	CZ	DA	DB	DC	DD	DE	DF	DG	DH	DI	DJ	DK	DL	DM	DN	DO	DP	DQ	DR	DS	DT	DU	DV	DW	DX	DY	DZ	EA	EB	EC	ED	EE	EF	EG	EH	EI	EJ	EK	EL	EM	EN	EO	EP	EQ	ER	ES	ET	EU	EV	EW	EX	EY	EZ	FA	FB	FC	FD	FE	FF	FG	FH	FI	FJ	FK	FL	FM	FN	FO	FP	FQ	FR	FS	FT	FU	FV	FW	FX	FY	FZ	GA	GB	GC	GD	GE	GF	GG	GH	GI	GJ	GK	GL	GM	GN	GO	GP	GQ	GR	GS	GT	GU	GV	GW	GX	GY	GZ	HA	HB	HC	HD	HE	HF	HG	HH	HI	HJ	HK	HL	HM	HN	HO	HP	HQ	HR	HS	HT	HU	HV	HW	HX	HY	HZ	IA	IB	IC	ID	IE	IF	IG	IH	II	IJ	IK	IL	IM	IN	IO	IP	IQ	IR	IS	IT	IU	IV	IW	IX	IY	IZ	JA	JB	JC	JD	JE	JF	JG	JH	JI	JJ	JK	JL	JM	JN	JO	JP	JQ	JR	JS	JT	JU	JV	JW	JX	JY	JZ	KA	KB	KC	KD	KE	KF	KG	KH	KI	KJ	KK	KL	KM	KN	KO	KP	KQ	KR	KS	KT	KU	KV	KW	KX	KY	KZ	LA	LB	LC	LD	LE	LF	LG	LH	LI	LJ	LK	LL	LM	LN	LO	LP	LQ	LR	LS	LT	LU	LV	LW	LX	LY	LZ	MA	MB	MC	MD	ME	MF	MG	MH	MI	MJ	MK	ML	MM	MN	MO	MP	MQ	MR	MS	MT	MU	MV	MW	MX	MY	MZ	NA	NB	NC	ND	NE	NF	NG	NH	NI	NJ	NK	NL	NM	NN	NO	NP	NQ	NR	NS	NT	NU	NV	NW	NX	NY	NZ	OA	OB	OC	OD	OE	OF	OG	OH	OI	OJ	OK	OL	OM	ON	OO															
																																																																																																																																																																																																																																																																																																																																																																																																																				NAME	kv	P MW	Q MVAR	LOAD = + MW	CAP = - MVAR	P MAX MW	P GEN MW	Q MAX MVAR	Q MIN MVAR	V HOLD PU	V MIN PU	REMOTE BUS NAME	kv	% SUPPLD

OWNER FIELD:  
Recommended for additions

ZONE FIELD:  
Required for additions, not required for modifications

= Required in most cases; blanks or zeros are unacceptable; exceptions are noted.

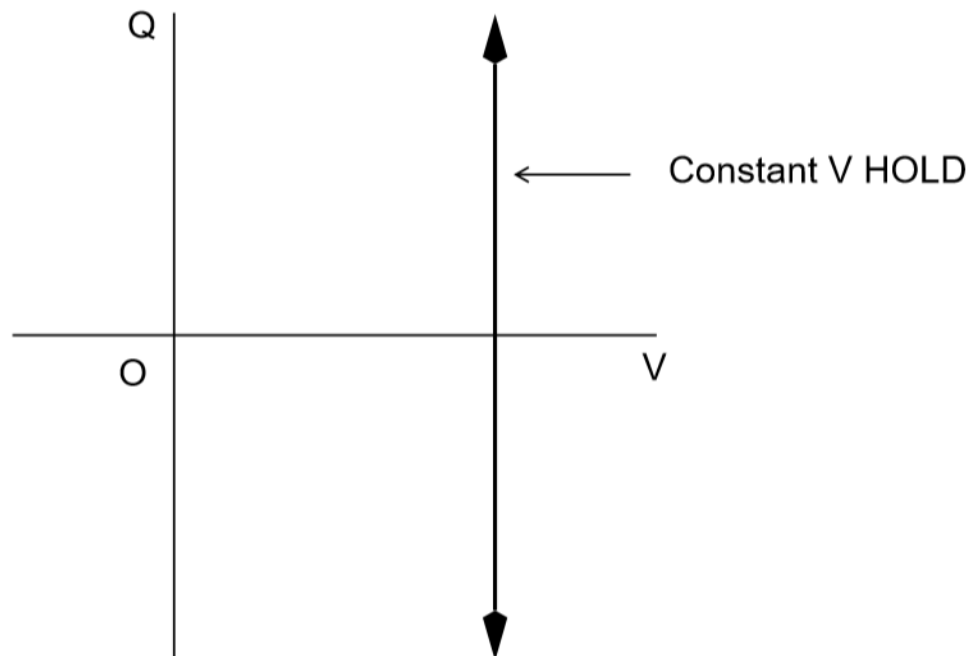
= Optional; blanks or zeros are acceptable.

= Not Applicable.

Fig. 2.5.11: BE Subtype Format

Table 2.5.11: Column Description for BE Bus Data

Column	ID Field	Format	Description
1-2	yes	A2	BE - Constant voltage bus
3	no	A1	Change code
4-6	no	A3	Ownership
7-14	yes	A8	Bus name
15-18	yes	F4.0	Base kV
19-20	no	A2	Zone
21-25	no	F5.0	Load MW
26-30	no	F5.0	Load MVAR
31-34	no	F4.0	Shunt Admittance Load in MW at base kV
35-38	no	F4.0	Shunt Admittance in MVAR
39-42	no	F4.0	P MAX
43-47	no	F5.0	P GEN
48-52	no	F5.0	Q MAX in MVAR
53-57	no	F5.0	Q MIN in MVAR
58-61	no	F4.3	V HOLD in per unit
62-65	no	F4.3	N/A
66-77	no	A8,F4.0	N/A
78-80	no	F3.0	N/A



Bus voltage will be held at a constant value implying infinite Q resources.

Fig. 2.5.12: Q-V Curve for BE Subtype

### 2.5.11 AC Bus Data (BF)

This is a special purpose bus type used to assist the Newton-Raphson solution convergence. The BF type behaves as a BE bus until the P<sub>net</sub> converges to the Newton-Raphson solution. Then it functions as a B- type. This feature is useful to bias a solution toward a more feasible voltage.

### 2.5.12 AC Bus Data (BG)

#### Application

This bus type is typically used to maintain the voltage at a remote bus (subtype BC).

This subtype may also be used for local control. For this application, the bus would maintain its own voltage. In this case, it would differ from a BQ bus only by the voltage limit. BG has  $V_{min} V_{max}$ ; BQ has  $V_{hold}$ .

#### Bus Characteristics

Real ( $P$ ) power is held constant. This applies to generators, load and shunt devices. However, it is not required to have generation (P GEN) at this bus.

Reactive ( $Q$ ) load and shunt are held constant for this bus type.

Reactive ( $Q$ ) generation is variable.

This bus type requires reactive limits to be entered in the Q MAX and Q MIN fields. The reactive limits on the subtype BG bus are used to maintain a specified voltage at a remote bus. If the remote bus voltage cannot be held with the available BG bus reactance, voltage control stops at either Q MAX or Q MIN.

A specific amount of reactive generation (Q SCHED) cannot be requested.

This bus type uses V MAX and V MIN limits. If these fields are blank, global voltages are used as defaults. The voltage on the BG bus must be between V MIN and V MAX when controlling a remote bus. If not, remote voltage control will be disabled.

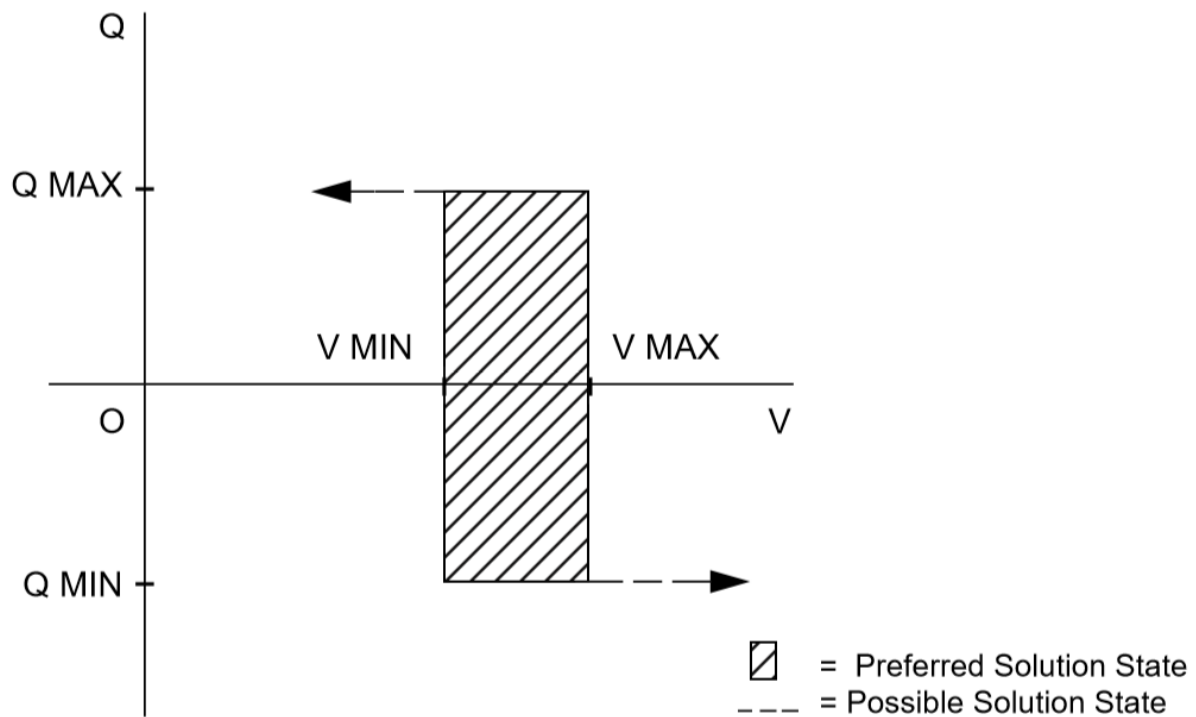
If this bus type is being used to control another bus, the REMOTE BUS and % VARS SUPPLIED fields should be used. The REMOTE BUS may not be the system swing bus or another subtype BG bus. The % VARS SUPPLIED field is used to allow the voltage control of a remote bus, to be distributed among more than one BG bus.

If the % VARS SUPPLIED is left blank, the program computes these values based upon the percent of total VARS supplied by the bus.

If the bus is controlling itself, its own name must appear in the REMOTE BUS field.







Shaded area depicts region of active control. Voltage and Q are variable within constraints shown.

Fig. 2.5.14: Q-V Curve for BG Subtype

## 2.5.13 AC Bus Data (BQ)

### Application

This subtype is used to hold the bus voltage to a specified value within reactive limits.

### Bus Characteristics

Real power (P GEN) is held constant. This applies to generators, load, and shunt devices.

Reactive load (Q MVAR) load is held constant for this bus type.

Reactive (SHUNT MVAR) shunt is variable. The amount of shunt reactance added by the program can vary from 0 to Qshunt, depending on the amount needed to maintain desired bus voltage.

Reactive (Q) generation is variable.

This bus type requires adjustable reactive generation or shunt to perform as intended. If neither is available, the bus functions as a bus type B-blank.

A specific amount of reactive generation (Q SCHED) cannot be requested.

Since this bus type is attempting to maintain its own voltage, a V HOLD entry is required on the record. If the voltage cannot be held at the desired level, using the reactive capability of the bus, the desired voltage will be violated and reactive will be held at the Q MAX or Q MIN limit. Note that V HOLD is not required for modifications.

B Q T C O W M R	OWNER										NAME	KV	Z O N E	LOAD										SHUNT		P MAX MW	P GEN MW	Q MAX MVAR	Q MIN MVAR	HOLD PU	V MIN PU	REMOTE BUS										%S V P L D																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
	P MW					Q MVAR								LOAD = +	CAP = -	NAME					KV																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
	1	2	3	4	5	6	7	8	9	0				1	2	3	4	5	6	7		8	9	0	1							2	3	4	5	6	7	8	9	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1

Table 2.5.13: Column Description for BQ Bus Data

Column	ID Field	Format	Description
1-2	yes	A2	BQ - Constant voltage within Q limits
3	no	A1	Change code
4-6	no	A3	Ownership
7-14	yes	A8	Bus name
15-18	yes	F4.0	Base kV
19-20	no	A2	Zone
21-25	no	F5.0	Load MW
26-30	no	F5.0	Load MVAR
31-34	no	F4.0	Shunt Admittance Load in MW at base kV
35-38	no	F4.0	Shunt Admittance in MVAR
39-42	no	F4.0	P MAX
43-47	no	F5.0	P GEN
48-52	no	F5.0	Q MAX
53-57	no	F5.0	Q MIN
58-61	no	F4.3	V HOLD
62-65	no	F4.3	N/A
66-77	no	A8,F4.0	N/A
78-80	no	F3.0	N/A

## 2.5.14 AC Bus Data (BS)

### Application

This subtype designates the system swing or slack bus. The generators at the swing bus supply the difference between the specified power flowing into the system at the other buses and the total system output plus losses. Thus, real and reactive power are determined as part of the solution for this subtype.

Every power flow case must have a minimum of one swing bus. In addition, each isolated AC system must have its own swing bus. The maximum numbers of swing buses allowed for a single power flow case is ten.

### Bus Characteristics

Real ( $P$ ) load is held constant. Both real ( $P$ ) generation and shunt are variable. The P GEN field is updated to the base case value. The P MAX field is used for reporting purposes only.

Reactive ( $Q$ ) load is held constant for this bus type. Reactive ( $Q$ ) shunt is variable. The amount of shunt reactance added by the program can vary from 0 to Qshunt, depending on the amount needed to maintain desired bus voltage.

Reactive ( $Q$ ) generation is variable.

Reactive constraints are allowed for this bus type. These quantities are entered in the Q MAX and Q MIN fields. If reactive constraints are imposed, “unscheduled reactive” may be added by the program to maintain the bus voltage.

A specific amount of reactive generation can be requested, in place of reactive constraints. This is implemented by entering a 0 in the Q MIN field and the desired amount of reactive generation in the Q SCHED field. Again, “unscheduled reactive” may be added by the program to maintain the bus voltage.

The BS bus record requires an entry in the V HOLD field.

The V MIN field is used to specify the angle of the swing bus for this application. It should be noted that an implied decimal point exists between columns 64 and 65. For example, an angle of 3.7 degrees can be specified with a 3 in column 64 and a 7 in column 65.

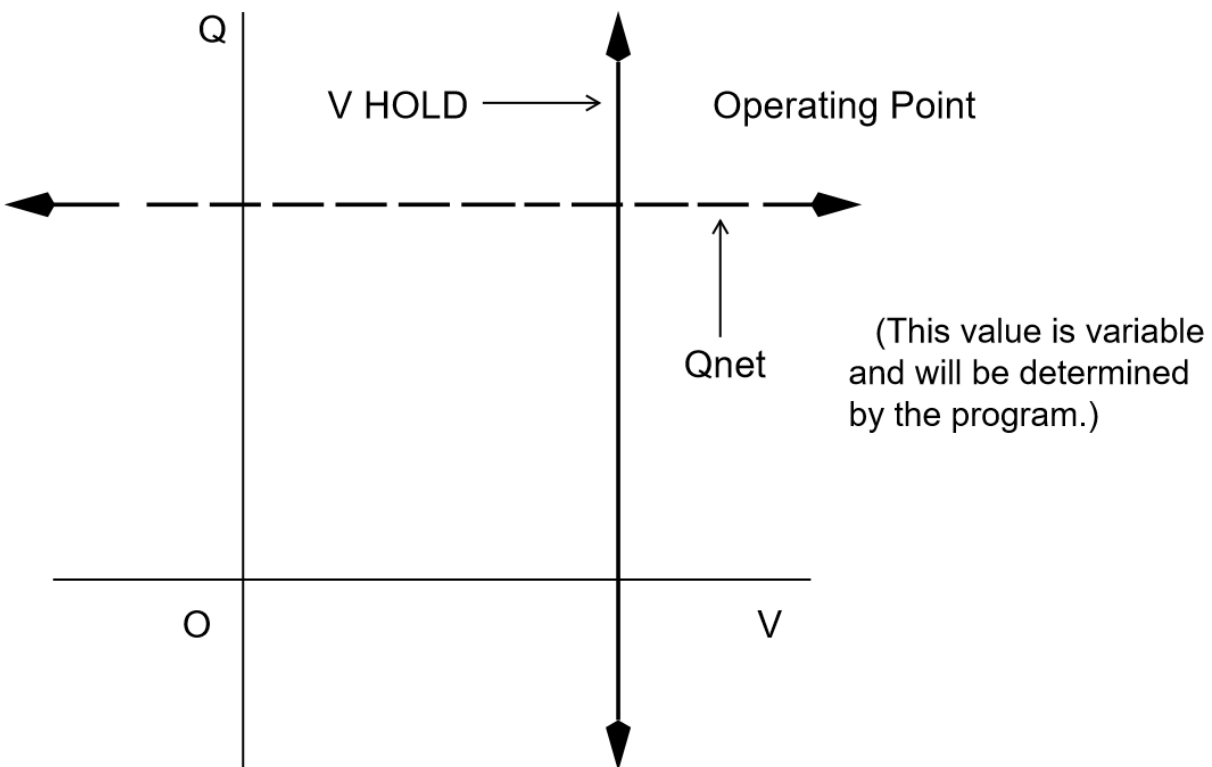
Fig. 2.5.16: Q-V Curve for BQ Subtype

[illegible]

Fig. 2.5.17: BS Subtype Format

Table 2.5.14: Column Description for BS Bus Data

Column	ID Field	Format	Description
1-2	yes	A2	BS - System swing or slack bus
3	no	A1	Change code
4-6	no	A3	Ownership
7-14	yes	A8	Bus name
15-18	yes	F4.0	Base kV
19-20	no	A2	Zone
21-25	no	F5.0	Load MW
26-30	no	F5.0	Load MVAR
31-34	no	F4.0	Shunt Admittance Load in MW at base kV
35-38	no	F4.0	Shunt Admittance in MVAR
39-42	no	F4.0	P MAX
43-47	no	F5.0	P GEN
48-52	no	F5.0	Q MAX
53-57	no	F5.0	Q MIN
58-61	no	F4.3	V HOLD
62-65	no	F4.1	Voltage angle (blank implies zero degrees)
66-77	no	A8,F4.0	N/A
78-80	no	F3.0	N/A



Bus Voltage and Phase Angle Are Held.

Fig. 2.5.18: Q-V Curve for BS Subtype



Table 2.5.15: Column Description for BT Bus Data

Column	ID Field	Format	Description
1-2	yes	A2	BT - LTC transformer controlled bus
3	no	A1	Change code
4-6	no	A3	Ownership
7-14	yes	A8	Bus name
15-18	yes	F4.0	Base kV
19-20	no	A2	Zone
21-25	no	F5.0	Load MW
26-30	no	F5.0	Load MVAR
31-34	no	F4.0	Shunt Admittance Load in MW at base kV
35-38	no	F4.0	Shunt Admittance in MVAR
39-42	no	F4.0	P MAX
43-47	no	F5.0	P GEN
48-52	no	F5.0	Q SCHED
53-57	no	F5.0	QMIN must be blank or zero for QSCHED to apply
58-61	no	F4.3	V HOLD
62-65	no	F4.1	N/A
66-77	no	A8,F4.0	N/A
78-80	no	F3.0	N/A

## 2.5.16 AC Bus Data (BV)

### Application

This subtype maintains the bus's net reactive ( $Q_{net}$ ) power flow as long as the bus voltage does not violate the user specified voltage range.

### Bus Characteristics

Real ( $P$ ) power is held constant throughout the entire solution. This applies to generators, load, and shunt devices.

Reactive ( $Q$ ) load and shunt are also held constant.

Reactive ( $Q$ ) generation is normally constant. Although this bus type actually has infinitely adjustable reactive limits, the program attempts to hold  $Q_{net}$  constant. However, if either of the voltage limits are violated,  $Q_{net}$  is changed to hold that limit. If any additional reactive generation is added by the program, it will be referred to as “unscheduled reactive” in the program output file.

A specific amount of reactive generation can be requested. This is accomplished by entering a zero (0) in the Q MIN field and the desired amount of reactive generation in the Q SCHED field.

Reactive constraints are not allowed for this type of bus. If reactive limits are entered in the Q MAX and Q MIN fields, they will be ignored.

The BV bus record requires entries in the V MAX and V MIN fields. The program's solution voltage will be within the range of V MAX to V MIN, regardless of how much reactive is required. If voltage attempts to rise above V MAX, additional negative reactive ( $-Q_{shunt}$ ) is added to bring the voltage down to V MAX. Also, if the voltage is below V MIN, additional reactive ( $+Q_{shunt}$ ) is added until the bus voltage has reached V MIN.

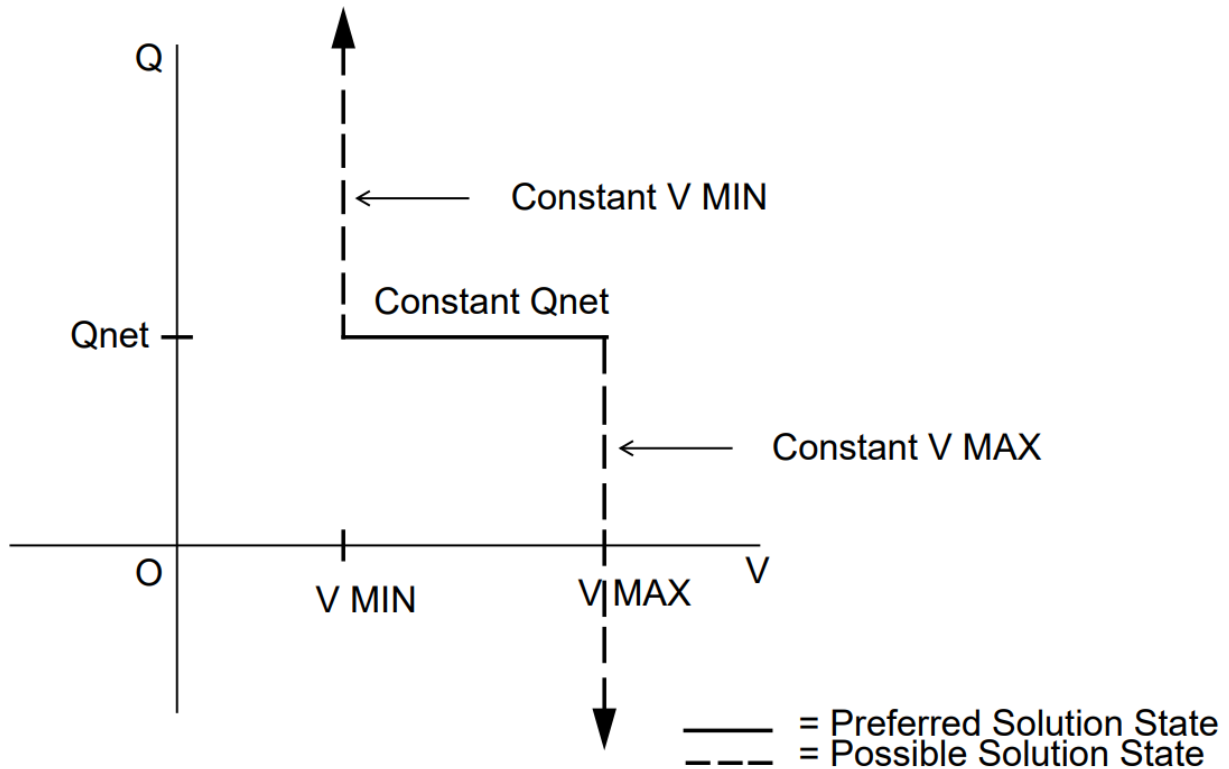


Fig. 2.5.19: BV Subtype Format

Column	ID Field	Format	Description
1-2	yes	A2	BV - Constant $Q$ within $V$ limits
3	no	A1	Change code
4-6	no	A3	Ownership
7-14	yes	A8	Bus name
15-18	yes	F4.0	Base kV
19-20	no	A2	Zone
21-25	no	F5.0	Load MW
26-30	no	F5.0	Load MVAR
31-34	no	F4.0	Shunt Admittance Load in MW at base kV
35-38	no	F4.0	Shunt Admittance in MVAR
39-42	no	F4.0	P MAX
43-47	no	F5.0	P GEN
48-52	no	F5.0	Q SCHED or QMA
53-57	no	F5.0	Q MIN
58-61	no	F4.3	V MAX in per unit
62-65	no	F4.1	V MIN in per unit
66-77	no	A8,F4.0	N/A
78-80	no	F3.0	N/A

## Application

## 2.5. Record Formats



The preferred solution state is constant  $Q$ , but may change if voltage constraints are violated.

Fig. 2.5.20: Q-V Curve for BV Subtype

## Bus Characteristics

The real ( $G$ ) and reactive ( $B$ ) shunt fields hold the base case values of discrete reactance. These values may be updated by the solution. Real ( $P$ ) power is held constant. This applies to generators and loads.

Reactive ( $Q$ ) load is held constant. Reactive ( $Q$ ) generation is variable.

Reactive constraints are allowed for this bus type. These quantities are entered in the Q MAX and Q MIN fields. A specific amount of reactive generation (Q SCHED) is available only if QMAX = QMIN. This bus type uses V MAX and V MIN limits. If these fields are blank, global voltages are used as defaults. The voltage on the BG bus must be between V MIN and V MAX when controlling a remote bus. If not, remote voltage control will be disabled.

Reactive shunt ( $+/- Q$ ) is added in discrete blocks to maintain the desired bus voltage. The capacitive/inductive blocks of reactance are identified on the X (switched reactance) record. It should be noted that actual convergence is implemented with continuous susceptance control, then discretization occurs automatically. This means that exact voltage control may not be possible.

It should be noted that the program will attempt to select a discrete reactive step, which yields the highest voltage within the specified limits, so that losses can be minimized. This is the default (BPA) value, for the third level >MISC\_CNTRL Program Control Statement, X\_BUS option.

SUBTYPE	OWNER	NAME	KV	ZONE	LOAD		SHUNT		P MAX MW	P GEN MW	Q MAX MVAR	Q MIN MVAR	V MAX PU	V MIN PU	REMOTE BUS		%SPDRD		
					P MW	Q MVAR	LOAD = +	CAP = -							NAME	KV			
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
BX																			
BX																			
BX																			

OWNER FIELD:  
Recommended for additions

ZONE FIELD:  
Required for additions, not required for modifications

REMOTE BUS:  
Required when "BX" bus is used to control a remote bus.

= Required in most cases; blanks or zeros are unacceptable; exceptions are noted.

= Optional; blanks or zeros are acceptable.

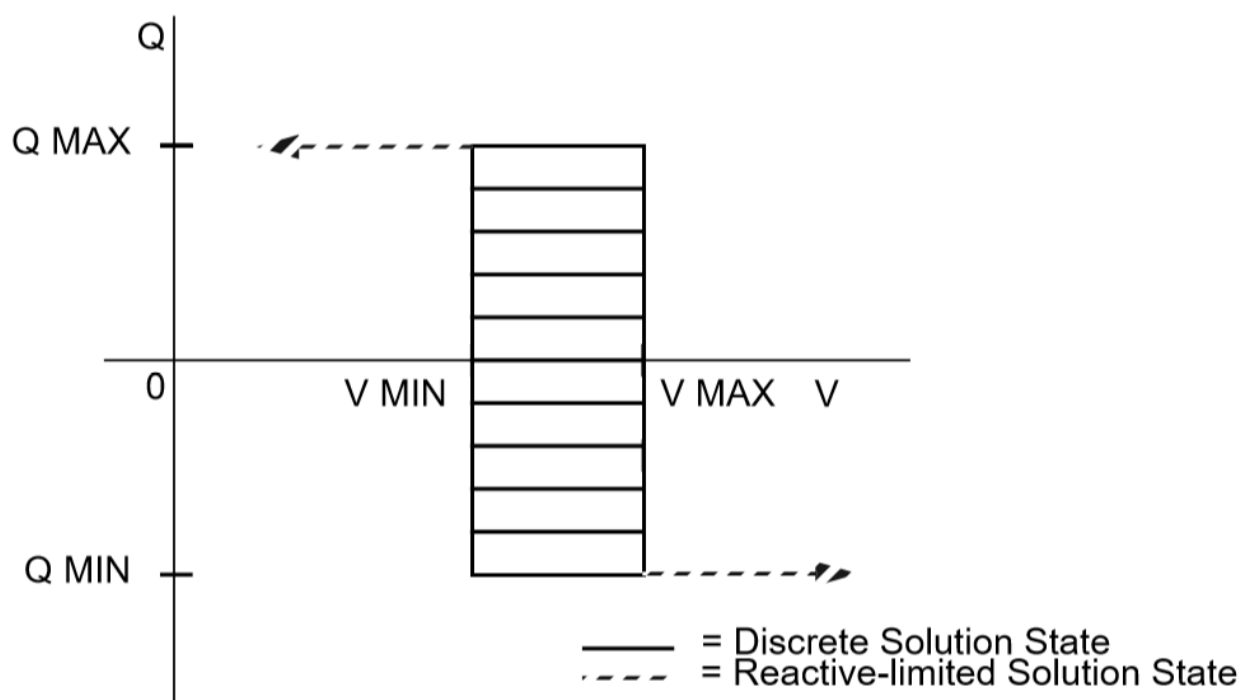
= Not Applicable.

Fig. 2.5.21: BX Subtype Format

**Note:** The value on the  $B_{shunt}$  field dictates the initial value

Table 2.5.17: Column Description for BX Bus Data

Column	ID Field	Format	Description
1-2	yes	A2	BX — Attempts constant V using switched Q
3	no	A1	Change code
4-6	no	A3	Ownership
7-14	yes	A8	Bus name
15-18	yes	F4.0	Base kV
19-20	no	A2	Zone
21-25	no	F5.0	Load MW
26-30	no	F5.0	Load MVAR
31-34	no	F4.0	Shunt Admittance Load in MW at base kV
35-38	no	F4.0	Shunt Admittance in MVAR
39-42	no	F4.0	P MAX
43-47	no	F5.0	P GEN
48-52	no	F5.0	Q SCHED or QMAX
53-57	no	F5.0	Q MIN
58-61	no	F4.3	V MAX in per unit
62-65	no	F4.1	V MIN in per unit
66-77	no	A8,F4.0	Controlled bus name and kV (self or remote). If blank, self is assumed.
78-80	no	F3.0	N/A



Enclosed area depicts region of active control. Voltage and Q are variable within constraints shown.

Fig. 2.5.22: Q-V Curve for BX Subtype



Table 2.5.18: Column Description for BD Bus Data

Column	ID Field	Format	Description
1-2	yes	A2	BD — Code for direct current (DC) bus, terminal of a DC line.
3	no	A1	Change code — see System Changes
4-6	no	A3	Ownership code
7-14	yes	A8	Bus name
15-18	yes	F4.0	Base kV
19-20	no	A2	Zone
24-25	no	I2	Bridges per ckt. — Number of valves in series per circuit.
26-30	no	F5.1	Smoothing reactor (mh) — Inductance of the smoothing reactor in millihenries.
31-35	no	F5.1	Rectifier operation (alpha min.) — Minimum firing angle in degrees as a rectifier.
36-40	no	F5.1	Inverter operation (alpha stop) — Maximum firing angle in degrees. Both inverter and rectifier buses have alpha. However, only the minimum alpha on the rectifier bus is used in the power flow. The remaining valves are required for the transient stability program in event of power reversals in the dc line.
41-45	no	F5.1	Valve drop (volts) — Valve voltage drop per bridge in volts.
46-50	no	F5.1	Bridge current rating (amps) — Maximum bridge current in amperes.
51-62	no	A8,F4.0	Commutating bus — Alphanumeric name in columns 51-58 and base kV in columns 59-62. This is on the AC system side of the commutating transformer bank

## 2.5.19 Multi-Terminal DC Bus (BM)

The multi-terminal DC system introduces flexibility in network configuration which is already present in the AC system. This DC scheme is a general extension of the two-terminal DC scheme. The converter modeling itself is unchanged, but the DC converter control is more flexible. All N-node DC systems must have N DC constraints. These are either converter DC voltage or DC power. At least one DC voltage constraint must be specified. It is permissible to constrain both DC voltage and power on the same node. The choice of voltage or power constraints on each converter is flexible.

It is possible to define a DC tap node. This node is passive only and is not a converter. Nevertheless, it implicitly constrains zero power on itself.

The distinction between rectifiers and inverters is very simple. Any converter's mode of operation is based upon the sign of the converter-calculated output power. The converter output power is positive for rectifiers and negative for inverters. Obviously, a DC tap node will have zero power.

The range of converter angle adjustments is determined by the converters's mode of operation.

$$\alpha_{min} \leq \alpha_{stop}$$

$$\gamma_0 \leq \alpha_{stop}$$

If an excessive number of DC constraints are specified, some superfluous power constraints will be omitted. If the DC system is unable to maintain the dc voltage constraints, the DC voltages will be changed to values realized by the actual commutator bus voltage and the converter angle limits.

It is permissible to model two-terminal DC networks with the multi-terminal type M formats. However, it is not permissible to mix two-terminal type D data with multi-terminal type M data on the same DC circuit. The two different types of DC data may coexist in the same case, but when both types are present, they must pertain to separate DC circuits.

[illegible]

Fig. 2.5.24: BM Subtype Format

Table 2.5.19: Column Description for BM Bus Data

Col- umn	ID Field	For- mat	Description
1-2	yes	A2	BM - Code for multi-terminal dc bus
3	no	A1	Change code - see System Changes
4-6	no	A3	Ownership code
7- 14ac	yes	A8	Bus name
15- 18	yes	F4.0	Base kV
19- 20	no	A2	Zone code
24- 25	no	I2	Number of bridges per DC circuit - (Number of converters serially connected)
26- 30	no	F5.1	Smoothing reactor inductance in mh
31- 35	no	F5.1	Minimum ignition delay angle ( $\alpha_{min}$ ) in degrees
36- 40	no	F5.1	Maximum ignition delay angle ( $\alpha_{stop}$ ) in degrees
41- 45	no	F5.1	Converter valve drop per bridge in volts
46- 50	no	F5.1	Maximum converter current in amps
51- 62	no	A8,F4.0	Commutator bus name and base kV of commutator.
63b	no	A1	Converter code (R1): R — Normal operation as a rectifier I — Normal operation as an Inverter M — Normal operation as an inverter with current margin Blank — A passive DC tap
64- 66	no	F3.1	Normal ignition delay angle ( $\alpha_N$ ) if a rectifier, or normal extinction angle ( $\gamma_N$ ) if an inverter, in degrees
67- 69	no	F3.1	Minimum ignition angle ( $\alpha_{min}$ ) if a rectifier, or minimum extinction angle ( $\gamma_0$ ) in degrees if an inverter
70- 75c	no	F6.1	Scheduled net converter DC output power in MW
76- 80d	no	F5.1	Scheduled converter DC voltage in kV

- A passive DC node has columns 24-80 all blank
- If the actual converter operation does not correspond to the converter code, subsequent swing studies will abort.
- If the DC power or voltage is not constrained, leave the corresponding field blank or enter a zero value.
- If the DC power or voltage is not constrained, leave the corresponding field blank or enter a zero value.



### 2.5.20 Delete Buses by Area (DA)

This command deletes all buses that reside in the area named in columns 4-13. Place a DA in the first two columns. Format of the input is shown below. This is a change record and must be preceded with a /CHANGES command or otherwise reside in a change set.

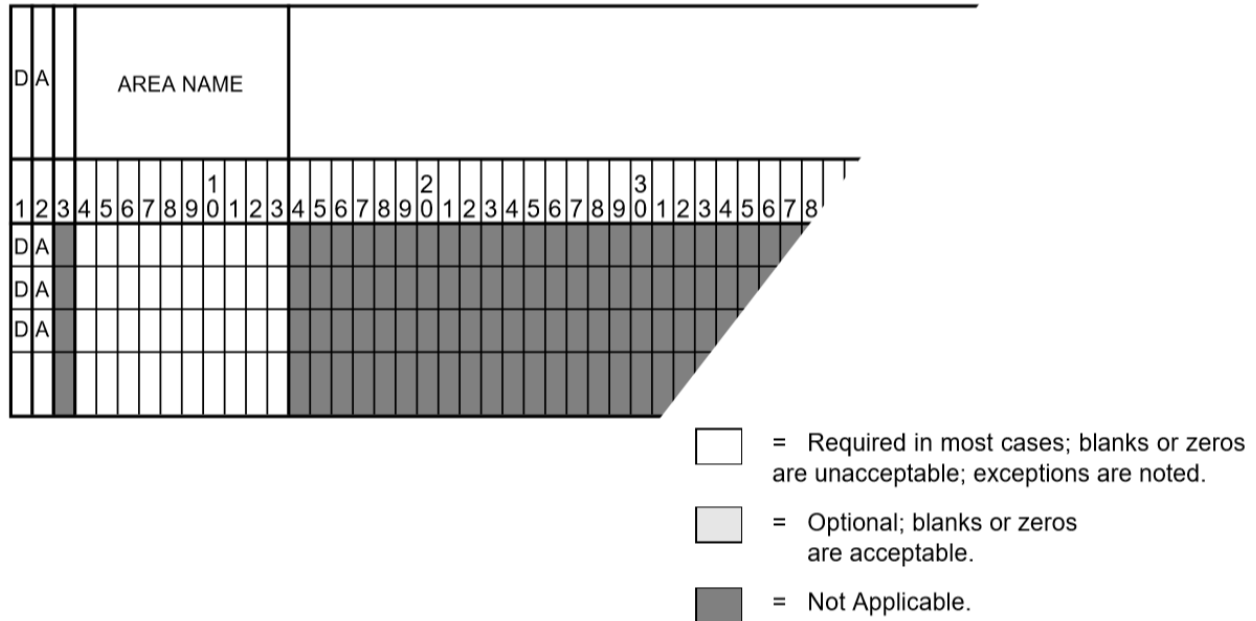


Fig. 2.5.25: Area Delete Input Format

Table 2.5.20: Column Description for Delete Buses by Area Format


Column	ID Field	Format	Content
1-2	yes	A2	Record type - DA
4-13	yes	A10	Area Name


### 2.5.21 Delete Buses by Zone (DZ)


This command deletes all buses that reside in the zone named in columns 4-5. Place a DZ in the first two columns. Format for input is shown below. This is a change record and must be preceded with a /CHANGES command or otherwise reside in a change set.

Table 2.5.21: Column Description for Delete Buses by Zone Format

Column	ID Field	Format	Content
1-2	yes	A2	Record type - DZ
4-5	yes	A2	Zone Name

 = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.

 = Optional; blanks or zeros are acceptable.

 = Not Applicable.

[illegible]

Fig. 2.5.27: Equivalent Branch Data Input Format

Table 2.5.22: Column Description for Equivalent Branch Data

Column	ID Field	Format	Description
1	yes	A1	Record type - E for equivalent
2	no	A1	Blank
3	no	A1	Change code - see System Changes
4-6	no	A3	Ownership code - Line and transformer losses will be summarized by ownership at end of final area summary.
7-14	yes	A8	Bus name 1
15-18	yes	F4.0	Base kV 1
19	no	I1	Tie line metering point flag for area tie lines. 1 in column 19 provides for metering at bus name 1 2 in column 19 provides for metering at bus name 2 Blank allows for program assumption as follows: Metering point will be identified (1) by location where line ownership differs from bus ownership or (2) when buses at end of tie line have same ownership, then the bus Name 1 will be the metering point.
20-27	yes	A8	Bus name 2
29-31	yes	F4.0	Base kV 2
32	yes	A1	Circuit identification
33	yes	I1	Section number for making an equivalent for series elements. (numeric)
34-37	no	F4.0	Total ampere rating for all lines represented by this record.
38	no	I1	Number of parallel circuits represented by this record, for information purposes only. The equivalent impedance is entered in columns 39-74 for lines with unequal legs.
39-50	no	2F6.5	Per unit R and X on base kV and base MVA.
51-62	no	2F6.5	Per unit G and B at bus name 1 end of line.
63-74	no	2F6.5	Per unit G and B at bus name 2 end of line.
75-77	no	A1,I2	<b>Energization Date - MYY</b>
<b>64</b>			<b>Chapter 2. Contents</b> {1,2,3,4,5,6,7,8,9,O,N,D} YY = last two digits of year

### 2.5.23 Scheduled Area Intertie (I)

Intertie is the power flowing between two areas. Scheduled Area Intertie are the values scheduled on the area bubble diagrams from which the net area interchange export is derived. The net export is simply the sum of all individual interties emanating from each area. Intertie I records permit the net area interchange schedules to be defined directly from the scheduled intertie quantities. New net area export is computed from these values; they override any scheduled net interchange on the AC control records.

A coding sheet and column descriptions follow.

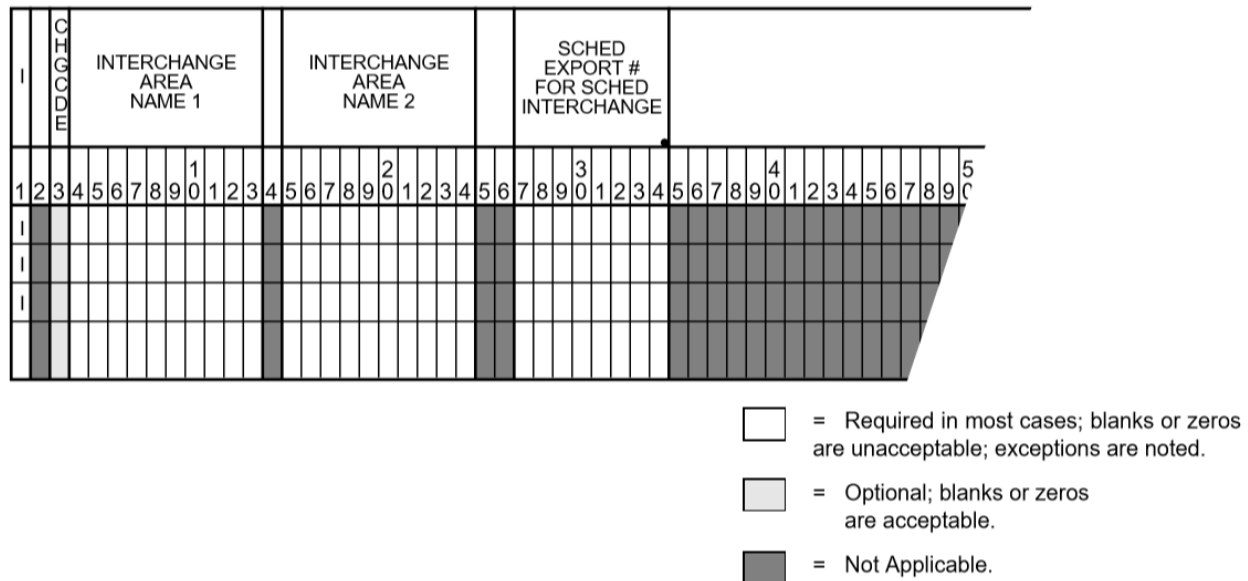


Fig. 2.5.28: Area Intertie Input Format

Table 2.5.23: Column Description for Scheduled Area Intertie

Column	ID Field	Format	Description
1	yes	A1	Record type - I
3	no	A1	Change code - see System Changes
4-13	yes	A10	Area 1
15-24	yes	A10	Area 2
27-34	no	F8.0	Area 1 - Area 2 export in MW. (Import will be negative.) Blanks are interpreted as 0.0 scheduled export

“Scheduled” is actually misapplied because no direct controls are available to regulate the intertie flow between two areas. Net area export can be controlled using area slack buses, but not interarea export, which requires additional but unavailable intertie slack buses. Therefore, without direct control of intertie, the scheduled flow can never be maintained. The term circulating flow is introduced to reconcile the discrepancy between the scheduled and the actual flow:

$$\text{Circulating flow} = \text{Scheduled flow} - \text{Actual flow}$$

Circulating flow is also a misnomer because it implies wasteful circulating power eddies within a network. If any circulating flow exists, it is introduced deliberately with the application of transformers (MVAR) or phase shifters (MW). Otherwise the flows will always be distributed optimally to minimize losses by simply following Kirchoff's voltage and current laws. The circulating flows are calculated in the interchange output.

### 2.5.24 Balanced Transmission Line Branch (L)

This record defines the identification and the electrical characteristics of a line, section of a line or series capacitor. The model assumes the form of a lumped, symmetric pi. The following identifies a branch item:

- Line type (L in this case).
- Bus 1 (name and base kV) and bus 2 (name and base kV).
- Circuit identification if more than one parallel branch exists.
- Section number (if appropriate).

Lines can be divided into equivalent series elements identified with unique section numbers. Section numbers need not be consecutive, but must be unique. Sections are presumed to be physically ordered such that the lowest numbered section is connected to bus 1 and the highest is connected to bus 2.

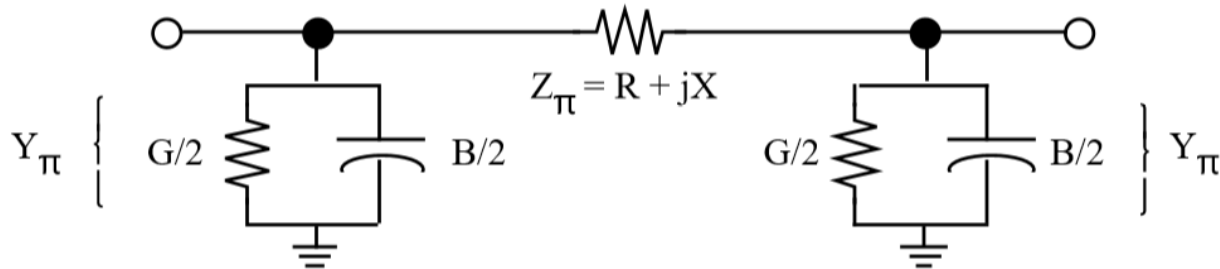


Fig. 2.5.29: Balanced Transmission Line Branch

The entries in **RATING** and **MILES** are used in output to flag overloaded lines and produce a MW-Miles listing by ownership and voltage class if requested.

The metering point (1 or 2) is used when the line spans two areas which are controlled. A simple comparison of the zones for each terminal bus will determine if that branch is a tie line. The metering point field determines which end of a line will represent the area boundary. The line losses are assigned to the other area. The entry in the **No. of CKTS** field (column 38) identifies the number of parallel branches represented by the branch item. A blank or zero is interpreted as one. This is for information purposes only.

The **DATE IN** and **DATE OUT** columns specify the expected energization and de-energization dates. These are used for descriptive purposes.

To simulate a bus tie or bus sectionalizing breaker normally closed, a line impedance of  $X = 0.00020 p.u.$  is used. This allows two sections to be connected or disconnected by adding or deleting this branch.

A sample coding sheet with column explanations follows.

L	ID	Q	G	H	C	OWNER	NAME 1										KV 1		METER	NAME 2										KV 2		C S E T T I O N	TOTAL CUR- RENT RATE AMP	# OF C K T	Z <sub>π</sub>													
																																			R							X						
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7												
L																																																
L																																																
L																																																

Z <sub>π</sub>										Y <sub>π</sub> /2										MILES	DESC DATA	DATE IN		DATE OUT		THERM RAT. AMPS	BOTL- NECK RAT. AMPS										
X					G/2					B/2					M	Y	M	Y																			
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8

☐ = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.  
☐ = Optional; blanks or zeros are acceptable.  
☐ = Not Applicable.

Fig. 2.5.30: Transmission Line Data Input Format

Table 2.5.24: Column Description for Transmission Line Data

Column	ID Field	Format	Description
1	yes	A1	Record type - L for branch
2	no	A1	Blank
3	no	A1	Change code - see System Changes
4-6	no	A3	Ownership code - Line and transformer losses will be summarized by ownership at end of final area summary.
7-14	yes	A8	Bus name 1
15-18	yes	F4.0	Base kV 1
19	no	I1	The line metering point for area tie lines. 1 in column 19 provides for metering at bus name 1 end. 2 in column 19 provides for metering at bus name 2 end. Blank allows for program assumption as follows: Metering point will be identified (1) by location where line ownership differs from bus ownership or (2) when buses at end of tie line have same ownership, then the bus name 1 will be the metering point.
20-27	yes	A8	Bus name 2
29-31	yes	F4.0	Base kV 2
32	no	A1	Circuit identification if more than one parallel branch exists.
33	no	I1	Section number for making an equivalent for series elements (numeric). Program assembles series elements in numerical order of section numbers (need not be consecutive).
34-37	no	F4.0	Total ampere rating for all lines.
38	no	I1	Number of parallel circuits represented by this record, for information purposes only. The equivalent impedance is entered in columns 39-62 for lines with equal legs.
39-44	no	F6.5	Per unit R at base kV and base MVA (normally 100).
45-50	no	F6.5	Per unit X at base kV and base MVA (normally 100).
51-56	no	F6.5	Per unit $G_{pi}/2$ at base kV and MVA (normally 100).



### 2.5.25 Two-Terminal DC Line (LD)

This record is used in conjunction with *Two-Terminal DC Bus (BD)* records; collectively they define the dc model. The dc line data contains pertinent information describing the electrical characteristics of the line, the scheduled dc power and voltage, and the initial firing (ignition) and extinction angles.

Distinction between the inverter and rectifier buses is made with the DC line record using the sign of the scheduled DC power. The direction of power flow is always from rectifier to inverter, and the criteria assumes positive values from bus 1 to bus 2 on the record. Thus, simple modifications in the line data permit power reversals to be modeled with a minimum of data changes. A sample DC line data coding form and column descriptions follow. See figure and table below.

L	S	C	H	G	C	O	W	CONVERTER 1								CONVERTER 2								TOTAL CUR- RENT RATE AMP	DC LINE PARAMETER																
								NAME								BASE KV									NAME								BASE KV								R
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5							
LD																																									
LD																																									
LD																																									

PARAMETERS																		I O R	DC LINE POWER (+ OR -)	RECT. VOLT	REC- TIFIER ( $\alpha_N$ )	IN- VER- TER ( $\gamma_0$ )	MILES	THERM RAT. AMPS	BOTL- NECK RAT. AMPS										
L									C																										
3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8

= Required in most cases; blanks or zeros are unacceptable; exceptions are noted.

= Optional; blanks or zeros are acceptable.

= Not Applicable.

Fig. 2.5.31: DC Line Data Input Format

Table 2.5.25: Column Description for DC Line Data

Columns	SD Field	Format	Descriptions
1-2	yes	A2	Record code - LD for DC line
3	no	A1	Change code - see System Changes
4-6	no	A3	Ownership code, same as on Bus record
7-14	yes	A8	Converter bus name 1 (conventionally the rectifier)
15-18	yes	F4.0	Base kV 1
20-27	yes	A8	Converter bus name 2 (conventionally the inverter)
29-31	yes	F4.0	Base kV 2
34-37	no	F4.0	I rating (amps) - Maximum DC line current in amperes.
38-43	no	F6.2	R (ohms) - DC line resistance, ohms.
44-49	no	F6.2	L (mH) - DC line inductance, millihenries
50-55	no	F6.2	C (uF) - DC line capacitance, microfarads.
56	no	A1	Inverter or rectifier control - Enter R for rectifier control or I for inverter control (point of DC line in which scheduled power is measured).
57-61	no	F5.1	Schedule DC power (MW) - Scheduled DC power in megawatts from converter 1 to 2 metered at the end indicated by I or R in column 56.
62-66	no	F5.1	Schedule DC line volts (kV) - at rectifier end of DC line.
67-70	no	F4.1	Rectifier ( $\alpha_N$ ) - Initial firing angle in degrees at rectifier.
71-74	no	F4.1	Inverter ( $\gamma_0$ ) - Minimum margin angle in degrees at inverter.
75-78	no	F4.0	Miles - Descriptive information only.
81-84	no	F4.0	Thermal rating in Amps
85-88	no	F4.0	Bottleneck rating in Amps

### 2.5.26 Multiterminal DC Line (LM)

This data is used in conjunction with *Multi-Terminal DC Bus (BM)* records. The line data together with the DC bus data define the DC converter terminals and interconnecting DC lines for a multiterminal DC network. This line data contains only the fields for some of the converter quantities which were included on the two-terminal DC format.

Two-terminal (type LD) and multiterminal (type LM) data may coexist within the same base, but cannot coexist on the same DC circuit.

The following shows the multiterminal DC line data coding form format. It is followed by explanations of its various columns.

[illegible]

Table 2.5.26: Column Description for Multiterminal DC Line Type LM

Column	ID Field	Format	Description
1-2	yes	A2	Record Code - LM for multiterminal DC line
3	no	A1	Change code - see System Changes
4-6	no	A3	Ownership code, same as on bus data record
7-14	yes	A8	DC bus name 1
15-18	yes	F4.0	Base kV 1
19	no	I1	Tie line metering point for area tie lines. 1 in column 19 provides for metering at bus name 1 end. 2 in column 19 provides for metering at bus name 2 end. Blank allows for program assumption as follows: Metering point will be identified (1) by location where line ownership differs from bus ownership or (2) when buses at end of tie line have same ownership, then the bus name 1 will be the metering point
20-27	yes	A8	DC bus name 2
29-31	yes	F4.0	Base kV 2
34-37	no	F4.0	I Rating (Amps) - Maximum DC line current in amperes
38-43	no	F6.2	R (Ohms) - DC line resistance in ohms
44-49	no	F6.2	L (mH) - DC line inductance in millihenries
50-55	no	F6.2	C (uF) - DC line capacitance in microfarads
71-74	no	F4.0	Miles - Descriptive information only
75-77	no	A1,I2	<b>Energization Date - MYY</b> M = {1,2,3,4,5,6,7,8,9,O,N,D} YY = last two digits of year
78-80	no	A1,I2	<b>De-energization Data - MYY</b> M = {0,1,2,3,4,5,6,7,8,9,O,N,D} YY = last two digits of year

### 2.5.27 Factor Change (P0, PZ, PN, PA, PB, PC, PD)

A specialized change record with a P in column 1 enables the loads and generation for all or part of the network to be changed on a multiplying factor basis. These records follow a /CHANGES statement and any Z records.

---

**Note:** These change record types were formerly called “Percentage Changes.” This is a change record and must be preceded with a /CHANGES command or otherwise reside in a change set.

---

Factor changes are performed before any other bus changes are made; therefore, any subsequent bus changes will be unaffected by factor changes. Separate fields permit both active and reactive generation and the active and reactive load to be changed at independent factor. The formula is:

$$NEWINJECTION = OLDINJECTION * (FACTOR)$$

A blank or 1.0 factor retains the present value of the injection while a factor of 2.0 will double the present value.

There are separate factors for  $P$  and  $Q$ . If the  $Q$  factor is blank, it will default to the  $P$  factor. For example, if only the  $P$  is to be changed by 0.95 and the  $Q$  left alone, set the  $P$  factor = .95 and  $Q$  factor = 1.0.

Six different subtypes are permitted. Different subtypes may be submitted together. (Their formats are shown on the following pages.)

Subtype 0 Factor change by ownerships (20 maximum). All AC bus and continuation bus loads and generation with the given ownership will be changed by the specified factors.

Subtype Z Factor change by zones (20 maximum). All AC bus and continuation bus loads and generation within the depicted zones will be changed by the specified factors.

Subtype N Factor change by zone on all nonindustrial loads and generation (20 maximum). All AC bus and continuation bus data except +A, +F, +I or +P continuation bus data within the designated zones will be changed by the specified factors. All zones may be changed with one record having a key zone 00 (zero-zero).

Subtype A Factor change on all loads and generation on all AC bus and continuation bus data (1 only).

Subtype B Factor change by ownerships (20 maximum). All constant impedance and constant current distributed (+ A01, + \*I, + \*P) loads with the given ownership will be changed by the specified factor.

Subtype C Factor change by zones (20 maximum). All constant impedance and constant distributed loads (+ A01, + \*I, + \*P) with the given zone will be changed by the specified factor.

Subtype D Factor change by zones (20 maximum). all non-industrial loads within the given zones and optional ownership will be converted into constant impedance and constant current loads.

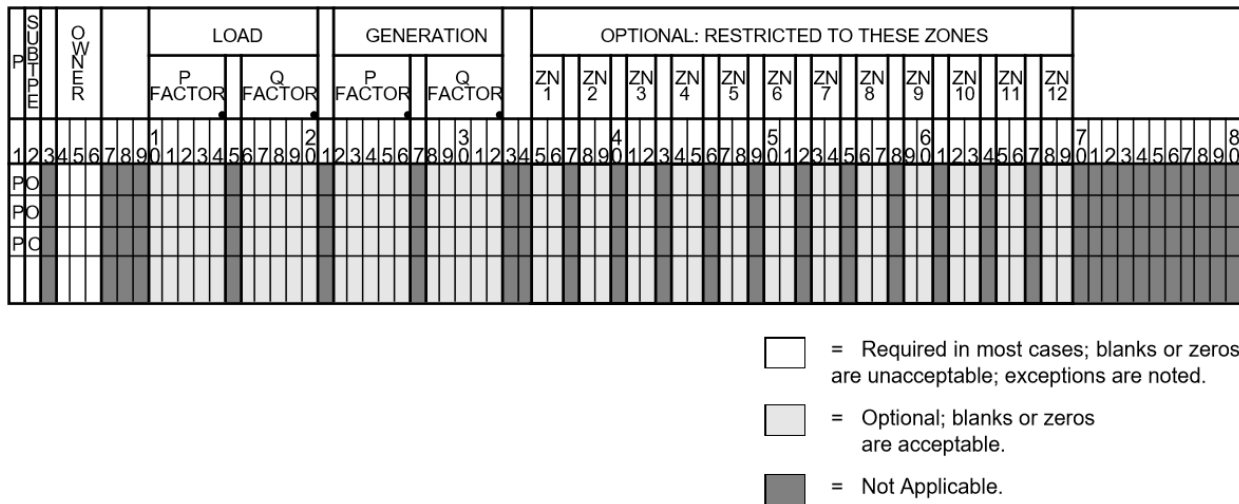


Fig. 2.5.32: Factor Change by Ownerships

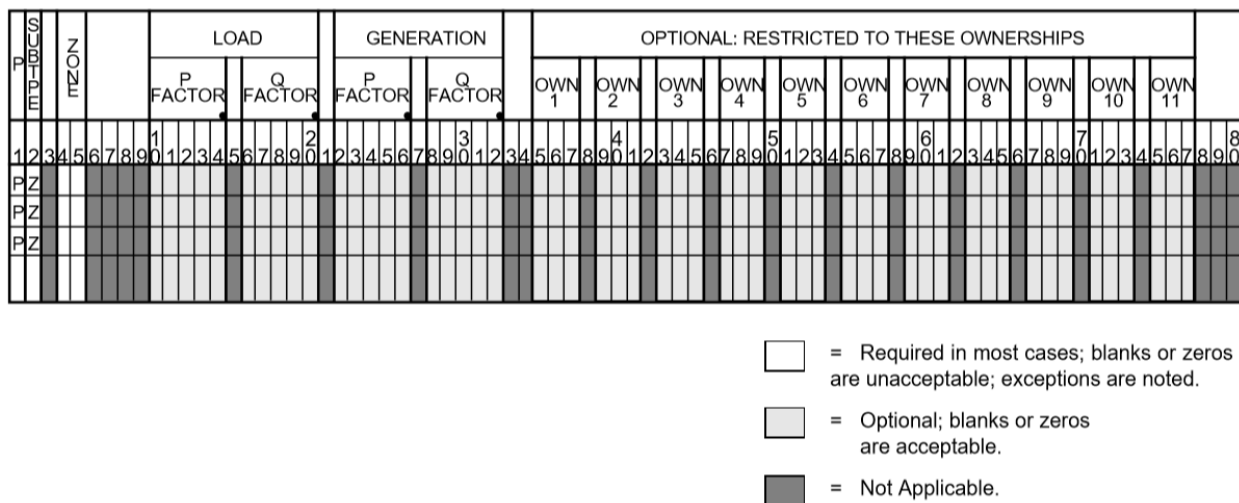







Fig. 2.5.33: Factor Change by Zones


 = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.

 = Optional; blanks or zeros are acceptable.

 = Not Applicable.

 = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.

 = Optional; blanks or zeros are acceptable.

 = Not Applicable.

SUB TYPE	ZONE	CURRENT CONSTANT										CONSTANT IMPEDANCE				OPTIONAL: RESTRICTED TO THESE ZONES																																			
		P FACTOR					Q FACTOR					P FACTOR		Q FACTOR		ZN 1	ZN 2	ZN 3	ZN 4	ZN 5	ZN 6	ZN 7	ZN 8	ZN 9	ZN 10	ZN 11	ZN 12																								
		1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12																
1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10		
PB																																																			
PB																																																			
PB																																																			

- ☐ = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.  
☐ = Optional; blanks or zeros are acceptable.  
☐ = Not Applicable.

Fig. 2.5.36: Factor Change by Ownerships of Constant Current and Impedance Loads

SUB TYPE	ZONE	CONSTANT CURRENT										CONSTANT IMPEDANCE				OPTIONAL: RESTRICTED TO THESE OWNERSHIPS																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
		P FACTOR					Q FACTOR					P FACTOR		Q FACTOR		OWN 1	OWN 2	OWN 3	OWN 4	OWN 5	OWN 6	OWN 7	OWN 8	OWN 9	OWN 10	OWN 11																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
		1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
PC																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					

- ☐ = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.  
☐ = Optional; blanks or zeros are acceptable.  
☐ = Not Applicable.

Fig. 2.5.37: Factor Change by Zones of Constant Current and Impedance Loads

SUB TYPE	ZONE	CONSTANT CURRENT										CONSTANT IMPEDANCE										OPTIONAL: RESTRICTED TO THESE OWNERSHIPS																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
		P FACTOR					Q FACTOR					P FACTOR					Q FACTOR					OWN 1	OWN 2	OWN 3	OWN 4	OWN 5	OWN 6	OWN 7	OWN 8	OWN 9	OWN 10	OWN 11																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
		1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
PD																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					

- ☐ = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.  
☐ = Optional; blanks or zeros are acceptable.  
☐ = Not Applicable.

Fig. 2.5.38: Factor Change by Nonindustrial Loads of Constant Current and Impedance Loads



Table 2.5.27: Column Description of Factor Change Record

Column		Format	Description
1	yes	A1	Record type - P factor change (P represents its former name Percentage Changes)
2	yes	A1	<b>Subtype</b> 0 for ownerships Z for all loads in selected zones N for nonindustrial loads in selected zones A for all loads and generatives B for constant current and constant impedance loads by owner-ship <b>C for constant current and constant impedance loads by zones</b>
10-14	no	F5.0	P factor (decimal fraction) for load
16-20	no	F5.0	Q factor (decimal fraction) for load
22-26	no	F5.0	P factor (decimal fraction) for generation or constant impedance loads
28-32	no	F5.0	Q factor (decimal fraction) for generation or constant impedance loads
Subtypes O and B - record			
4-6	yes	A3	Owner code
35-80	no	(A2,1X)	Optional list of zone codes separated by single blanks. Change will be restricted to these listed zones. If no list, all zones are assumed. Note: Subtype C is restricted to the following types of continuation buses: +A01, +A02, *P, *I.
Subtypes Z, C, and N			
4-5	yes	A2	Zone code
35-80	no	(A3,1X)	Optional list of ownership codes separated by single blanks. Change will be restricted to these listed owners, if no list, all owners are assumed. Note: Subtype C is restricted to the following types of continuation buses: +A01, +A02, *P, *I.
<b>2.5. Record Formats</b>			<b>27</b>

### 2.5.28 Reactive Capability Curves (QP, QX, QN)

Three records are required to define a curve: QP, QX, and QN. They may appear anywhere in the input stream although they normally are put immediately after the bus record to which the curve applies. Each curve applies only to the bus named.

#### Description

The generator capability curve model is a composite of two representations of a synchronous machine capability curve. The first model consists of a set of 1-14 points depicting a piece-wise linear representation of the Q-P characteristics; the second model consists of an optional, constant MVA representation.

As shown in the figure below, the generator capability curve model consists of a series of 1-15 points on the P-Q diagram and a constant MVA secant. Each point is defined by specifying a value for  $P$  followed by values for  $Q_{max}$  and  $Q_{min}$ . The constant MVA is specified with  $MMVA$  (Machine MVA), a leading powerfactor (positive reactive), and a lagging powerfactor (negative reactive). If  $MMVA$  is zero, the reactive capability curve is specified exclusively with  $P$ ,  $Q_{max}$ , and  $Q_{min}$  points

If the optional constant MVA representation is invoked, it takes precedence over any  $(P, Q_{max}, Q_{min})$  point. (Overlap is determined by  $P_{gen}$ .)

If the minimum absolute value for  $P$  is less than the first entered value ( $P_1$ ), then the model will set the values for  $Q_{max}$  and  $Q_{min}$  equal to  $Q_{max1}$  and  $Q_{min1}$ . For any point  $ABS(P_{gen})$  between  $P_1$  and  $MMVA$ , the model will linearly interpolate between the  $Q$  values for  $P_j$  just greater than and  $P_{j-1}$  just less than  $ABS(P_{gen})$ .  $P_{gen}$  greater than  $MMVA$  generates a fatal data error.

#### Processing

Before solution of the case, each BE, BG, BQ, BX, and BS bus is checked to see if a PQ curve is to be used to set its Q limits. If not, the  $Q_{min}$  and  $Q_{max}$  already stored are used, that is, those read from the bus record or calculated from a prior solution. If a curve is active, the values calculated using it replace those formerly stored. Original input values from the bus record are not saved.

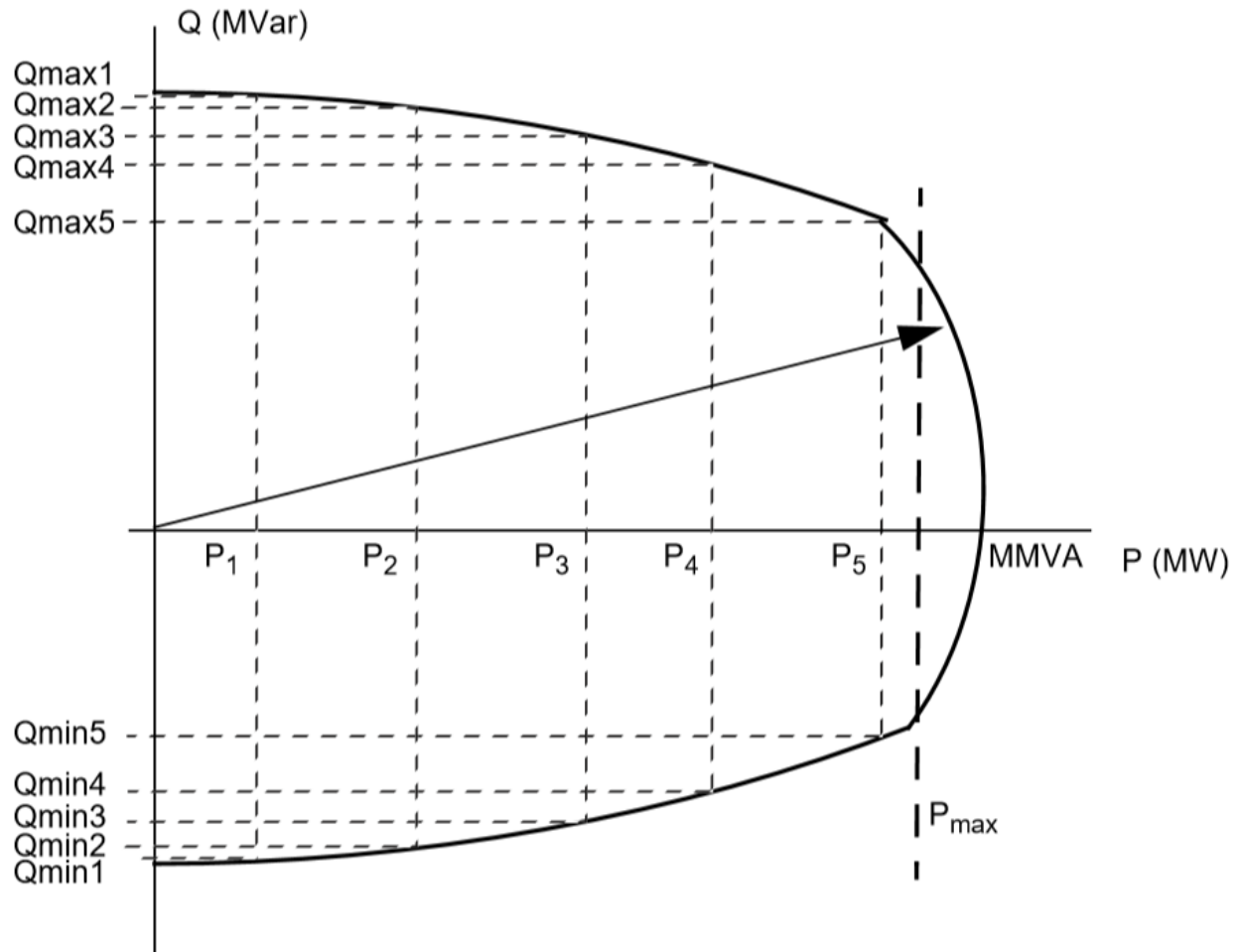


Fig. 2.5.39: Generator Capability Curve Model

Table 2.5.28: Column Description for Reactive Capability Curves

Column	ID Field	Format	Description
1-2	yes	A2	<b>Record Code -</b> QP for Pgen values (positive values only) QX for Qmax values (positive values) QN for Qmin values (negative values)
3	no	A1	<b>Change code - For QP record only:</b> D = Delete curve for this bus. M = Change p.u. code or activity flag. <i>This cannot be used to alter curve data.</i> To change curve data, enter a complete new set. It is not necessary to delete the curve first; new data will replace the old. Column 3 must be blank.
4-5	no	A2	Unit ID (for informational purposes)
6	no	A1	<b>Activity flag For QP record only:</b> Blank = Curve active (default when data is entered). * = Inactivate curve. A = Activate formerly inactive curve.
7-14	yes	A8	Bus name
15-18	yes	F4.0	Base kV
19-20	no	I2	Number of units. Total used is number of units times the values specified ( $P_{gen}$ , $Q_{max}$ , $Q_{min}$ )
21-25	no	F5.2	QP - Maximum MVA QX - Positive (leading) power factor for Qmax QN - Negative (lagging) power factor for Qmin
26-30	no	F5.2	Maximum Pgen (MW/unit). This is an optional hard limit, designating a maximum operating limit.
31-120	no	15F6.2	Up to 15 values for Pgen, Qmax, or Qmin depending on the card type. The values for Pgen can be in any order, but the related

[illegible]

- Fig. 2.5.40: Reactive Capability Curve QP Record

### 2.5.29 Regulating Transformer (R, RV, RQ, RP, RN, RM)

This record defines the range of adjustable taps or angles, identifies the adjustable tap side and specifies the desired control and quantity to be held.

The tap or angle specified on a T record determines the initial parameter setting. If this setting is not encompassed by the LTC tap range, the R record is temporarily deactivated in the solution routine with diagnostic messages. The default LTC control can also be enabled by the command:

(continues on next page)

(continued from previous page)

ON\_NPS (R,RQ,RN only)  
ON\_DCONL(commutating LTC transformers on)  
OFF

If an LTC transformer reaches a tap limit, the control is temporarily deactivated.

All regulating transformers have provision for either continuous or discrete tap positions. Discretization occurs automatically after a continuous solution is formed.

Several subtypes are available:

**Subtype Blank or Subtype V** This specifies LTC voltage control on either of the terminal buses. The controlled voltage is entered on the appropriate bus record. The terminal being controlled may be an subtype except G, X or S. If the controlled bus has local reactive control (subtypes E or Q within its limits) the LTC feature is temporarily deactivated. However, if both terminal buses of the transformer are simultaneously reactively controlled, the LTC transformer assumes a different control to minimize the var exchange between buses. This assumes most effective utilization of local VAR sources.

Subtype Q This specifies an LTC transformer that controls the VAR flow through itself. Positive controlled quantities are from bus 1 to bus 2. However, for simplicity, the control point is taken inside the equivalent pi from the fixed tap side to the variable tap side. The controlled VARs value is  $Q_{scheduled} + / - Q_{tolerance}$  when  $Q_{tolerance}$  is the MIN scheduled field and  $Q_{scheduled}$  is the MAX scheduled field.

Subtype P This subtype defines an LTC phase shifter that controls real power flowing through itself. Positive controlled quantities are from bus 1 to bus 2. However, for simplicity, the control point is taken inside the equivalent pi. The controlled power is  $P_{scheduled} + / - P_{tolerance}$  ( $P_{scheduled} = \text{MAX SCHED}$ ;  $P_{tolerance} = \text{MIN SCHED}$ ).

Subtype N This specifies an LTC transformer that provides constraints on the reactive power flow through itself. Ordinarily, it provides no control, but if its reactive flow limits are exceeded, it becomes a type Q LTC transformer and holds the MVAR flow within the inequality constraints assigned. The controlled flow is within  $Q_{min} \leq Q'_{max}$  ( $math := \text{MIN SCHED}$ ;  $Q_{max} = \text{MAX SCHED}$ ).

Subtype M This specifies an LTC phase shifter that provides constraints on the active power flow through itself. Ordinarily, it provides no control, but if its active power limits are exceeded, it becomes a type P LTC phase shifter and holds the MW flow within the inequality constraints assigned. The controlled real power flow is within limits  $P_{min} \leq P_{max}$  ( $P_{min} = \text{MIN\_SCHED}$ ;  $P_{max} = \text{MAX\_SCHED}$ ).

R	S	B	C	T	I	C	O	M	O	M	R	BUS 1								V	BUS 2								RMTE CNTRLD BUS								MAX PHASE SHIFT ANGLE	MIN PHASE SHIFT ANGLE	#T OA FPS	MAX / SCHED MW	MIN MW																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
												NAME				BASE KV					NAME				BASE KV				NAME				BASE KV									MX TAP	MN TAP	MAX / SCHED MVAR	MIN MVAR																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
												1	2	3	4	5	6	7	8		9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24										25	26	27	28	29	30	31	32																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	




-  = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.
-  = Optional; blanks or zeros are acceptable.
-  = Not Applicable.

Fig. 2.5.41: Regulating Transformer Data Input Input Format

Table 2.5.29: Column Description for Regulating Transformer Data

Column	ID Field	Format	Description
1	yes	A1	Record type - R for LTC and automatic phase-shifter data. A record type T must be in same system to provide full data required.
2	no	A1	Subtype - See regulating transformer and phase-shifter subtypes.
3	no	A1	Change code - see System Changes
4-6	no	A3	Ownership code
7-14	yes	A8	Bus name 1
15-18	yes	F4.0	Base kV 1
19	no	I1	Variable tap side if T_max and T_min cannot orient T_x. 0 - Low alpha is fixed 1 - Bus 1 is variable 2 - Bus 2 is variable
20-27	yes	A8	Bus name 2
29-31	yes	F4.0	Base kV 2
34-45	no	A8,F4.0	Controlled bus name and base kV
Multiple Use of columns 46-67 For subtypes blank, V, Q and N			
46-55	no	2F5.2	Maximum and minimum kV taps. Data must be entered.
56-57	no	I2	Total number of LTC taps. If blank, program assumes continuous action.
58-67	no	2F5.0	Scheduled MVAR flow (subtype Q) or maximum and minimum MVAR flow (subtype N) through transformer Metered at bus name 1 on this record.
For subtypes P and M			
46-55	no	2F5.2	Maximum and minimum angle in degrees. Data must be entered.
56-57	no	I2	Total number of phase shift positions available. If blank, program assumes continuous action.
58-67	no	2F5.0	Scheduled MW flow (subtype P) or maximum and minimum MW flow (subtype M) through transformer. Metered at bus name 1 on this record.
For all subtypes			
75-77	no	A1,I2	
<b>2.5. Record Formats</b>			<b>Energization Date - 83</b> <b>MY Y</b> M = {1,2,3,4,5,6,7,8,9,O,N,D} YY = last two digits





Table 2.5.30: Column Description for Series Compensated RANI Model

Column	ID Field	Format	Description
1-2	yes	A2	Record type - RZ
3	no	A1	<b>Change Code</b> Blank - add M - modify D - delete R - restore
7-14	yes	A8	Bus name 1
15-18	yes	F4.0	Base kV 1
19	no	I1	Variable tap side if T_max and T_min cannot orient T_x. 0 - Low alpha is fixed 1 - Bus 1 is variable 2 - Bus 2 is variable
20-27	yes	A8	Bus name 2
29-31	yes	F4.0	Base kV 2
32	yes	A1	ID
33	yes	I1	SECTION
34	no	A1	<b>TYPE 1, 2, or 3</b> TYPE 1 - Control Pc using Xij TYPE 2 - Control V using Xij TYPE 3 - Control V using Bis
35-39	no	F5.0	Pcmax(MW)
40-44	no	F5.0	Pcmin(MW)
45-48	no	F4.0	Irate(amps)
49-54	no	F6.5	Xijmax(p.u.)
55-60	no	F6.5	Xijmin(p.u.)
61-66	no	F6.5	Bismax (p.u.)
67-72	no	F6.5	Bismin (p.u.)

### 2.5.31 Transformer Data (T, TP)

This record is applied to two-winding transformers and phase shifters. An equivalent pi representation depicts the transformer admittance in series with an ideal transformer. The electrical data is specified in terms of the transformer data which is defined as follows:

$R$  Equivalent resistance due to copper loss.

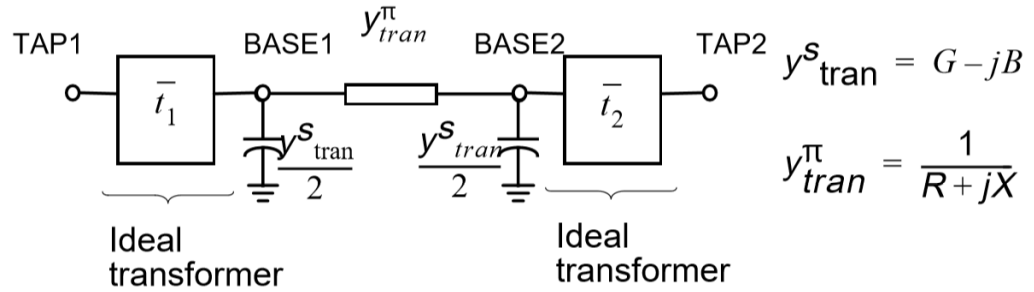
$X$  Leakage reactance.

$G$  Equivalent core loss conductance.

$B$  Magnetizing susceptance (always assumed negative; any sign is overridden).

Transformer taps are specified as fixed values for each voltage level or variable (LTC) taps with control over voltage, real power or reactive power. Variable tap transformers are defined with the addition of a regulating transformer data record (R) described in *Regulating Transformer (R, RV, RQ, RP, RN, RM)*.

The following three assumptions are made:



Transformers:  $t_1 = \frac{\text{TAP1}}{\text{BASE1}}$   $t_2 = \frac{\text{TAP2}}{\text{BASE2}}$

Phase shifters:  $t_2 = \epsilon^{\frac{J \text{TAP1}}{57.2957795}}$   $t_2 = 1$

Fig. 2.5.43: Transformer Data

- Nominal base kV of transformer is identical to that of the bus.
- Nominal base MVA of transformer is the same as the system base MVA.
- $R$ ,  $X$ ,  $G$  and  $B$  are evaluated on the nominal base and not on the tap setting.

The base kV is used to calculate the equivalent pi. Complications arise when step-up/step-down transformers are represented as sections in passive node sequence; there is no means to identify the base of the intermittent terminals. This combination is therefore illegal and will abort with fatal diagnostics.

A transformer is identified by subtype blank and a viable entry for TAP 2. A phase shifter is identified either by a subtype P or by a blank field for TAP 2. However, the program always adds the subtype P for distinction and convenience in data changes. The ANGLE is TAP 1 interpreted in degrees. If TAP 2 is blank, a nominal tap is presumed. A phase shifter assumes a pure voltage rotation and an optional transformation. Increasing the ANGLE will decrease the power flowing from bus 1 and bus 2.

Four MVA ratings are used to flag overloaded transformers: nominal, thermal, emergency, and bottleneck. The latter three are “extended ratings”, which supersede nominal ratings. Other entries are similar to the type L record. A sample of the format and explanations follow.

T	SUBTYPE	OWNER	NAME 1										KV 1	METER	NAME 2										KV 2	SECTION ID	TOTAL MVA RATE	# OF CKT	Z pi												
																													R					X							
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	
T																																									
T																																									
T																																									

Y pi										PHASE SHIFT DEG				DATE IN		DATE OUT		THERM RAT MVA	EMERGENCY RAT MVA	BTTL NECK RAT MVA				
G					B					TAP 1 KV	TAP 2 KV	MTH	YR	MTH	YR									
8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2

☐ = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.  
☐ = Optional; blanks or zeros are acceptable.  
☐ = Not Applicable.

Fig. 2.5.44: Transformer Data Input Format

Table 2.5.31: Column Description for Transformer Data

Column	ID Field	Format	Description
1	yes	A1	Record type - T for transformer or phase shifter
2	no	A1	Blank or P
3	no	A1	Change code - see System Changes
4-6	no	A3	Ownership code - Line and transformer losses will be summarized by ownership at end of final Area Summary.
7-14	yes	A8	Bus name 1
15-18	yes	F4.0	Base kV 1
19	no	I1	Tie line metering point flag for area tie lines. 1 in column 19 provides for metering at bus 1 end. 2 in column 19 provides for metering at bus 2 end. Blank allows for program assumption as follows: Metering point will be identified (1) by location where line ownership differs from bus ownership or (2) when buses at end of tie line have same ownership, then bus 1 will be the metering point.
20-27	yes	A8	Bus name 2
29-31	yes	F4.0	Base kV 2
32	yes	A1	Circuit identification
33	yes	I1	Section number for making an equivalent for series elements (numeric)
34-37	no	F4.0	Total MVA rating for all transformers represented by this record.
38	no	I1	Number of parallel transformer banks represented by this record, for information purposes only. The equivalent impedance is entered in columns 39-62.
39-44	no	F6.5	Per unit impedance R through transformer from bus 1 to bus 2 on the system base MVA for both windings.
45-50	no	F6.5	Per unit impedance X through transformer from bus 1 to bus 2 on the system base MVA for both windings.
51-56	no	F6.5	Per unit G of iron losses on the system base MVA.
57-62	no	F6.5	Per unit B magnetizing current on the system base

### 2.5.32 Switched Reactance (X)

Normally shunt capacitor or shunt reactor installations are represented in power flow programs as fixed MVAR values. However, there frequently is a need to represent voltage controlled capacitor schemes. This program allows for voltage controlled shunt device installations through use of a switched reactance record type X.

This data record must be used with a type BX bus record. The data on the X record identifies blocks of discrete shunt susceptance available for reactive control. The blocks may be either inductive (negative) or capacitive (positive). If both types exist, negative blocks must be given first. Values are given in MVAR at the rated base kV; the actual MVAR is dependent on the voltage. Discrete reactance units are also referred to as statics.

The discrete segments are specified in steps and increments. Steps defines the number of switchable susceptance units. The MVAR value of each unit is given in increments. Normally, each unit has a unique MVAR value. In general, when both reactive and capacitive blocks coexist, two switching lists are given. The reactors are switched first in the following order: unit 1, unit 2,..., unit k, where k is the unit number of last reactor. Similarly, the capacitors are switched next in the following order: unit k+1, unit k+2,..., unit (last). When reactors are switched in, all capacitors are removed and vice versa. Units are sequentially disconnected in the reverse order. For example, the following one line diagram depicts a capacitor installation requiring discrete switching.

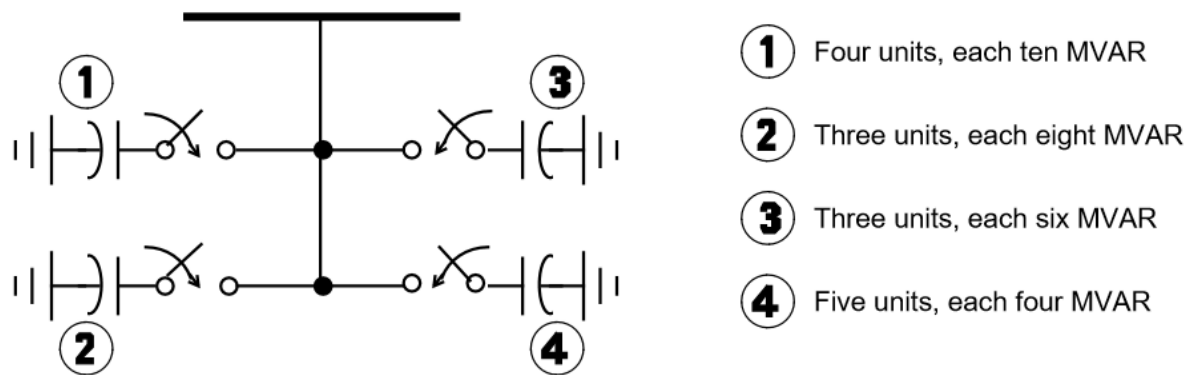


Fig. 2.5.45: Capacitor Installation Requiring Discrete Switching

For this example the sequential operation would be as follows. Increment 1, 10 MVAR, would be added up to the number of steps specified (4). If the bus voltage is below the scheduled value after a total of 40 MVARs have been added to the bus, increment 2 would be added to the number of steps specified, and so on until the scheduled voltage is achieved. These statics would be disconnected in a similar fashion, but in the reverse order, to reduce the bus voltage.


Shunt susceptance on BX records is ignored when X records are present. Any shunt susceptance listed on any following + records is considered fixed and is therefore exempt from the above switching.


The format for inputting switched reactance data follows; and then descriptions of the various columns are given.

---

**Note:** Negative MVARs must be listed first

---

 = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.

 = Optional; blanks or zeros are acceptable.


 = Not Applicable.

Fig. 2.5.46: Switched Reactance Data Input Format

Column	ID Field	Format	Description
1	yes	A1	Record code - X for switched reactance
2	no		Blank
3	no	A1	Change code - see System Changes
4-6	no	A3	Ownership code
7-14	yes	A8	Bus name 1. This is the bus to which the switched reactances are connected. Must be a type BX bus.
15-18	yes	F4.0	Base kV 1
21-28	yes	A8	Bus name 2. Controlled bus name - This bus may be local or remote.
30-32	yes	F4.0	Base kV 2
Increment 1:			
33	no	I1	Number of steps - Number of repetitions (from 1 to 9) to be performed using the value given in columns 34-38.
34-38	no	F5.0	MVAR - Value at base kV of each step of Increment 1.
Increment 2:			
39	no	I1	Number of steps - Number of times the value given in columns 40-44 will be connected. (Will not begin until Increment 1 is completed.)
40-44	no	F5.0	MVAR - Value at base kV of each step of Increment 2.
Increment 3:			
45	no	I1	Number of steps - Number of times the value given in columns 46-50 will be connected. (Will not begin until Increment 2 is completed.)
46-50	no	F5.0	MVAR - Value at base kV of each step of Increment 3.
Increment 4:			
51	no	I1	Number of steps - Number of times the value given in columns 52-56 will be connected. (Will not begin until Increment 3 is completed.)
52-56	no	F5.0	MVAR - Value at base kV of each step of Increment 4.
967-80	no	Add data for Increments 5 through 8 in a similar fashion.	

**Chapter 2. Contents**

### 2.5.33 Zone Rename (Z)

This is a specialized change record which permits the bus zones to be permanently renamed. This record has a Z in column 1, followed by a maximum of 15 pairs of old zone names/new zone names. Additional Z records may follow to change a maximum of 150 zones. See table below.

Several old zones may be consolidated into a single new zone name. However, an old zone may not be segregated into two more new zones.

Any Z record must be the first change record encountered. The zones are immediately renamed. Following these records, additional system changes may be entered in the normal way, and all data are assumed to conform with the new zone names.

Z	ZONE NAMES																																									
	OLD		NEW		OLD		NEW		OLD		NEW		OLD		NEW		OLD		NEW		OLD		NEW		OLD		NEW		OLD		NEW		OLD		NEW							
	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0		
Z																																										
Z																																										
Z																																										

= Required in most cases; blanks or zeros are unacceptable; exceptions are noted.

= Optional; blanks or zeros are acceptable.

= Not Applicable.

- ☐ = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.
- ☐ = Optional; blanks or zeros are acceptable.
- ☐ = Not Applicable.

Fig. 2.5.47: Zone Rename Data Input Format

Table 2.5.33: Column Description for Zone Rename

Column	ID Field	Format	Description
1	yes	A1	Record type - Z
2-3	no	A2	Blank
4-5	no	A2	Old zone name - any blank zone to be changed must appear in this column. Otherwise, blank "old zones," terminate the scan.
6-7	no	A2	New zone name
8	no	A2	Blank
9-10	no	A2	Old zone name
11-12	no	A2	New zone name
13	no	A1	Blank
14-15	no	A2	Old zone name
16-17	no	A2	New zone name
18-77	no	•	Repeat of the above format sequence: one blank followed by two columns for old zone name and two more columns for new zone name.

### 2.5.34 Area Rename (ZA)

This is a specialized change record that permits area names to be changed. Each ZA record renames one area.

The new name must be unique. If the new name exists, the rename is ignored. Thus, Area Rename cannot be used to consolidate areas.

[illegible]




-  = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.
-  = Optional; blanks or zeros are acceptable.
-  = Not Applicable.

Fig. 2.5.48: Area Rename Data Input Format



Column	ID Field	Format	Description
1-2	yes	A2	Record type - ZA
3			NA
4-13	no		A10 Old area name
14-15			NA
16-25	no	A10	New area name
26-80			NA

This is a specialized change record that permits bus names and base kV's as well to be altered. Each ZB record renames one bus.

[illegible]

- Fig. 2.5.49: Bus Rename Data Input Format

Column	ID Field	Format	Description
1-2	yes	A2	Record type - ZB
3-6			NA
7-14	no	A8	Old bus name
15-18	no	F4.0	Old base kV
19-20			NA
21-28	no	A8	New bus name
29-32	no	F4.0	New base kV
33-80			NA

## 2.6 Power Flow Control (PFC)

### 2.6.1 Overview

This section describes the batch Power Flow Control (PFC) language and its syntax, commands and subcommands. Command entries follow the PFC description in alphabetical order. The table below helps you turn quickly to a specific command entry. The table also gives you a quick description of all of the commands.

Each command entry explains the meaning of the command and gives its syntax. Some commands have subcommands, which are also described. Many entries have additional discussion, and some have examples, particularly where a command's usage may not be immediately obvious.

### 2.6.2 The bpf Control Language

The bpf Power Flow Control language (PFC) consists of a sequence of program control statements, each of which in turn consists of commands, subcommands, keywords, and values. All statements have a reserved symbol in column 1 to identify a command or subcommand.

Every statement is scanned, and each command or subcommand found is compared with a dictionary in the program to find the relevant instructions. With the exception of the identifier in column 1 of each statement, PFC is free-form. All statements must be in the PFC file.

PFC has three levels of control, which are identified by one of three identifiers in column one.

1. The left parenthesis ( identifies the top (or process) level of control. Only four commands are valid here — (POWERFLOW, (NEXTCASE, and (STOP or (END.
2. The slash / identifies the second (or command) level of control. Many commands are valid here, and they are listed and described in this chapter. Commands generally enable or disable output options, define parameters needed for the process, etc. Subprocesses are major operations involving considerable processing and additional data. Only optional IPF processes are requested with these commands.
3. The right angle bracket > identifies the third (or subcommand) level of control. A few commands have subcommands associated with them. These subcommands are described in the associated command entries. These subcommands act as qualifiers for the second-level commands.

In addition to the foregoing syntactic units, a command enabling a microfiche option is available. Its control symbol is the left square bracket ([)

Almost every PFC statement fits one of the following formats, and the few that do not are very similar.

---

**Note:** Spaces can be used for readability. Commas are used to separate syntactic units such as a list of values or keyword/value assignments.

---

Most statements fit on one line, but some extend over multiple lines. These exceptions are noted. When used, put a hyphen (-) where you want to break and continue the command parameters starting in or after column 2 of the next line, column 1 must be blank.

Each general format is followed by an example.

- A simple command with no keywords or values:

/ command
/ REDUCTION

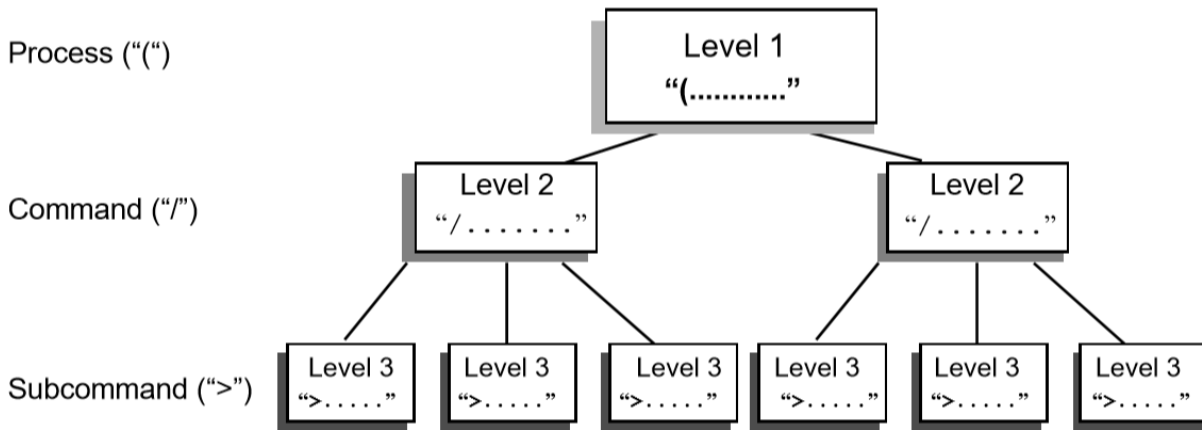


Fig. 2.6.1: Hierarchical Levels of PFC Statements

- A command assigned a simple keyword. (This is a “telescoped” syntax available for some commands.)

```
/ command = keyword
/ AI_CONTROL = CON
```

- A command followed by a comma with a keyword. (This is a “telescoped” syntax available for some commands.)

```
/ command , keyword
/ F_INPUTLIST, NONE
```

- A command followed by a comma with a value assigned to a keyword.

```
/ command , keyword = value
/ P_ANALYSIS_RPT, LEVEL = 4
```

- A command followed by a comma with multiple values assigned to a keyword. Note optional continuation with hyphen (-).

```
/ command , keyword = value , value , value , ...
/ P_INPUT_LIST, ZONES=NA,NB,NC,ND,NE,NF, -
    NG,NH,NI,NJ,NK
```

- A command followed by a comma with multiple value/keyword assignments.

```
/ command , keyword = value , keyword = value , ...
/ MERGE_OLD_BASE, SUB_SYSTEM_ID = AREA-1, OLD_BASE = TESTDC.BAS
```

- A command followed by a data record(s).

```
/ command data record
/ NETWORK_DATA
B    GEN1 HI    230  2  -0.0  -0.0  0.0  0.0      -0.0 -0.0 -0.0
B    GEN2 HI    230  1  -0.0  -0.0  0.0  0.0      -0.0 -0.0 -0.0
B    GEN3 HI    230  2  -0.0  -0.0  0.0  0.0      -0.0 -0.0 -0.0
```

- A subcommand followed by a comma with multiple comma-separated values.

```
>subcommand, value, value, value ,...  
>SAVE_ZONES,NA,NB,NC,ND,NE,NF,NG,NH,NI,NJ,NR
```

- A subcommand followed by a data record on the next line.

```
>subcommand  
data record  
>SAVE_AREAS  
A AREA 2
```

### 2.6.3 Special Characters

Two special characters are available to document the control stream or to improve readability.

- A period (.) in column 1 of a record identifies a command comment and the record will be ignored by the processing. It is used to document a PFC file or to improve readability. This comment is only visible in a listing of the PFC file or in the editor used to create it.
- The underscore symbol \_ has no syntactic significance and may be used freely to punctuate a word for visual readability.

---

**Note:** The hyphen or minus sign “-” and the underscore “\_” symbol are different characters! Thus, P\_O\_W\_E\_R\_F\_L\_O\_W is the same as POWER\_FLOW which is the equivalent of POWERFLOW. OLD\_BASE is the same as OLDBASE but not the same as OLD-BASE, etc.

---

### 2.6.4 Default Convention

All default values for a command are listed on the first line in the command descriptions. Various keywords are listed below the default values. Default values have been selected to satisfy a majority of users; therefore, their use is to invoke exceptions to standard conventions. Once a default value has been enabled, it remains in force for the duration of the process. There is one exception to this:

/ P\_INPUT\_LIST

After the first case has been processed, P\_INPUT\_LIST is set to NONE. This conforms to the default philosophy of selecting all options that fulfill a majority of requirements.

### 2.6.5 Microfiche Control Statement

```
` `[ FICHE,COPIES = n ]` `  
`` ( FICHE,COPIES = n ) ``
```

This command requests “n” copies of microfiche listings to be made. If it is omitted, the fiche file is not saved. If “n” is zero or omitted, no copies are made. When it is used, this control must be first in the job stream.

## 2.6.6 Level 1 PFC Commands

### ( POWERFLOW )

This command initiates the processing of the network which is defined with subsequent commands and subcommands.

### ( NEXTCASE )

This is the same as ( POWERFLOW ) except that the base network to be processed is the current network. Changes are expected; otherwise, the same network is processed again with the same data and controls in memory from the previous case. ( NEXT\_CASE ) cannot be the first command in a program control file.

### ( END ) or ( STOP )

This stops the execution of the IPF program.

Each network is processed with a ( POWERFLOW ) or ( NEXTCASE ) command. The first must always be ( POWERFLOW ). Several cases may be concatenated (stacked) in the following format:

```
``( POWERFLOW )`` statement for case 1
``( POWERFLOW )`` statement or ``( NEXTCASE )`` statement for case 2
.
.
.
``( POWERFLOW )`` statement or ``( NEXTCASE )`` statement for case n
``( STOP )``
```

The following control statement and the optional keywords that go with it identify the OLD\_BASE file, optionally perform miscellaneous temporary changes to OLDBASE, set solution parameters, and solve the resultant network.

```
(POWERFLOW CASEID = <casename>, PROJECT = <projname>)
```

casename is a user-assigned 10-character identification for the case. projname is a user-assigned, 20-character identification for the project or study to which this case applies. (No blanks are allowed; use hyphens instead.)

The following statement is used if the Powerflow solution is to be run starting with data and controls from the previous base case in residence.

```
( NEXTCASE, CASEID = <casename>, PROJECT = <projname> )
```

Note that / OLD\_BASE is not used with a ( NEXTCASE ) statement since a base data file is already in residence.

## 2.6.7 Level 2 and 3 PFC Commands

Each Level 2 statement starts with a slash (/) in the first position.

After the slash are keywords and/or values separated by a comma (,). Specific values are assigned to the keywords in the following format:

keyword = value

When a keyword is requesting a list, for example, a zone list, the list may be continued on the next record by leaving column 1 of that record blank or by putting a comma in column 1 and continuing the list.

Level 3 statements consist of subcommands that specify keyword values for second-level commands only. Each subcommand for level 3 statements starts with the right angle bracket (>) in column 1. After the right angle bracket are keywords and/or values separated by commas (,). Most often, specific values are assigned by following a keyword with an equal sign (=) and then the desired value.

## 2.6.8 PFC Commands

The rest of this chapter discusses all the PFC commands, in alphabetical order. Each command entry includes the details of syntax and usage. The more involved commands show examples of use. Refer to the table below to locate a PFC command quickly.

In the format statement for each command, the keywords and parameter values are all vertically aligned in the same column. The top row is the default value. Alternate value assignments such as ON or OFF are identified by the appropriate symbols and have the syntax `keyword=value`.

Required text is shown in UPPER-CASE while parameter values specified by the user are printed in lower-case and usually enclosed by angle brackets, thus, `<list>`. Angle brackets are omitted when they may cause confusion with the Level 3 control symbol.

The optional underscore symbol (`_`) may be used to break up words for visual readability. The computer will read the words as though they were not broken.

Table 2.6.1: PFC Commands

Command Name	Description
AGC	Emulates automatic generation control.
AI_LIST	Species detail in area interchange listing.
ANALYSIS_SELECT	Selects analysis reports for printing or microche.
BRANCH_DATA	Species a master branch and bus data le for base case.
BUS_SENSITIVITIES	Calculates system response to capacitor switching operations.
CHANGE_BUS_TYPE	Disables voltage control in system.
CHANGE_PARAMETERS	Perturbs parameters for start of new solution.
CHANGES	Species system data change records.
COMMENT	Species comment records.
COMMON_MODE_ANALYSIS	Analyzes the results from a series of common mode cases in the format of the OUTAGE_SIMUL.
F_ANALYSIS_RPT	Species report of zones or owners for microche.
F_INPUT_LIST	Lists input data on microche.
F_OUTPUT_LIST	Lists output data on microche.
GEN_DROP	Balances generation drop by picking up generation.
HEADER	Species header information for reports.
INCLUDE_CONTROL	Species a le for control commands.
LINE_EFF	Denes minimum percent line loading for report.
LINE_SENSITIVITIES	Determines line sensitivity by controlling LTC and AI_CONTROL.
%LOAD_DISTRIBUTION	Converts constant power, etc., into user-specied MVA, etc.
LOSS_SENSITIVITIES	Gives information about system losses.
MERGE_OLD_BASE and	MERGE_NEW_BASE Extracts information from two subsystems to create a new system.
MVA_BASE	Changes base MVA to an assigned MVA.
NETWORK_DATA	Species bus and branch data.
NEW_BASE	Denes the le name for a new case.
OI_LIST	Lists ownership interchange.
OLD_BASE	Species a previously existing solved case as the new case to start with.
OUTAGE_SIMULATION	Simulates line outages, load dropping, generator outages and rescheduling.
OVERLOAD_RPT	Species overload parameter limits for report.
P_ANALYSIS_RPT	Creates an analysis report.
P_INPUT_LIST	Lists input data on paper.
P_OUTPUT_LIST	Lists output on paper.
REBUILD	Rebuilds all data tables from current OLDBASE le.
REDUCTION	Reduces a network.
RPT_SORT	Sorts output data of solved network.

continues on next page

Table 2.6.1 – continued from previous page

Command Name	Description
SAVE_FILE	Creates various output files, including the SIF (Stability Interface File).
SOLUTION	Enables solution options and post-solution processes.
SORT_ANALYSIS	Controls sort order for analysis listings.
TRACE	Monitors data to aid data verification.
TRANSFER_SENSITIVITIES	Causes analysis of transfer sensitivities.
TX_EFF	Compares total and core transformer losses.
USER_ANALYSIS	Generates custom analysis listings

## 2.6.9 AGC

This command emulates automatic generation control (AGC) in the solution algorithm. Under AGC, real power excursions on several generators from base values are allocated in proportion to their total excursion. This in effect distributes the slack bus real power excursions to a set of selected units. The slack bus excursion, which drives AGC, may be either a system slack bus or an area slack bus.

The individual AGC units are identified with type B (bus) records which follow the / AGC command. columns (1:18) correspond with the original format. Beyond column 18, data is free field.

```
B <bus_name,base kV> Pmin=<##>, Pmax=<##>, Pgen=<##>, %=<##>
```

where

**Pmin** Minimum generation in MW. Default value is 0.0.

**Pmax** Maximum generation in MW. Default is Pmax, which is specified on the bus record.

**Pgen** Base generation in MW, which is used to compute the excursions. Default is scheduled or actual MW from the base case.

**%** Percentage. The default allocates% in proportion to Pmax

A maximum of 24 AGC units may be specified. One of the units must be a system or area slack bus. Usually, AGC schemes converge faster than non-AGC. The exception occurs when Pmin or Pmax limits are hit and some readjustment occurs.

## General Description

An example illustrates the concept. In Case 1, there are two generators, GEN1 and GEN2, with initial and final values shown in the table below.

Table 2.6.2: Values Without AGC

Bus	Initial P	Final P	Excursion
GEN1 (slack)	1000	1442	442
GEN2	1000	1000	0
TOTAL	2000	2442	442

In Case 2, we apply AGC with 50% on each machine. Presuming that losses are unchanged (for simplicity), the initial and final values are shown in the table below.

Table 2.6.3: Values With AGC

Bus	Initial P	%	Final P	Excursion
GEN1 (slack)	1000	50	1221	221
GEN2	1000	50	1221	221
TOTAL	2000	100	2442	442

**Notes and Restrictions** A maximum of 24 generators are permitted. One of the generators must be a system slack bus or an area interchange bus. Recall that the dynamics which drive AGC comes from slack bus P excursions.

If any unit hits a limit, the remaining active units redistribute their percentages and continue AGC control.

The results are summarized in the listing AGC Control. This listing is controlled with / ANALYSIS\_SELECT command.:

```
/ ANALYSIS_SELECT
> SUM%VAR
```

If area interchange control is ON, all AGC units should reside in the same area. Violations of this rule are flagged with warning diagnostics. AGC control will obscure the change in slack bus power shown in the tie line Summary of Area Interchange. The true slack bus effects within the area would be the aggregate effects of all AGC units. The area interchange summary obscures this effect.

When / AGC's and / GEN\_DROP coexist, / AGC operates with a higher priority. In actuality, the two should not coexist.

The validity of AGC can be verified in the analysis summary AGC Control. In normal conditions, the scheduled and actual percentage participation should be equal. If these quantities are not equal, it is usually because Pmax or Pmin limits have been hit. In this instance, a comment appears.

Actual % / Sched % = \*\*\*\*.\*

All of the active units should have an individual ratio

## 2.6.10 AI\_LIST

This command controls the level of detail in the area interchange listing.:

```
/ AI_LIST = FULL
```

FULL is the default. The options are:

FULL Area interchange matrix, Area slack bus summary, and tie line flows.

MATRIX Area interchange matrix.

TIELINE Tie line flow summary

## 2.6.11 ANALYSIS\_SELECT

This selects individual analysis reports for printing or microfiche. It supersedes / F\_ANALYSIS and / P\_ANALYSIS. Unlike these commands which select groups of reports according to their "level" the / ANALYSIS\_SELECT command selects reports individually.

A solitary / ANALYSIS\_SELECT command defaults all analysis listings to no print/no fiche status.

Printing and/or microfiche are enabled with the commands: > FICHE and > PAPER. These commands independently restrict the contents of FICHE or PAPER reports to subsets of Zones, Ownerships or Areas.



The desired analysis reports are individually selected using > commands containing abbreviated report names, e.g., > UNSCH.

Each > (report) command accepts an optional F or P qualifier. This will restrict the selected report to Fiche or Print respectively. If neither appear, both F and P are presumed to be selected. For example, > UNSCH, P will print the unscheduled reactive report.

A special option exists on the > LINEFF report. Its entirety is:

```
> LINEFF, SORT = BUS_NAME, OUTPUT=filename, FIELD_WIDTH = 132,F,P
                                     80,F,P
>
      VOLTAGE,
      OWNER,
      ZONE
```

All quantities are optional.

SORT Controls sorting by bus kV\_name, owner\_name, or zone name.

OUTPUT Copies a duplicate report to the named file.

FIELD\_WIDTH Controls the report width or the named file.

The following is a full list of the / ANALYSIS\_SELECT command set.:

```
/ ANALYSIS_SELECT
> FICHE,ZONES=<zone1,...>,AREAS=<area1,...>,OWNERS=<owner1,...>
> PAPER,ZONES=<zone1,...>,AREAS=<area1,...>,OWNERS=<owner1,...>
> USERAN - User-defined analysis listing. (Used with / USER_ANALYSIS.)
> UNSCH - Buses With Unscheduled Reactive.
> LOSSOWN - Total System Generations and Loads by Owner.
> SYSTEMZONE - System Generations,Loads,Losses and Shunts by Zones.
> UVOV - Undervoltage-Overvoltage Buses.
> LINELOAD - Transmission Lines Loaded Above xxx.x% of Ratings or with more than 30
↳degrees of electrical angle.
> TRANLOAD - Transformers Loaded Above xxx.x% of Ratings.
> TRANEX - Transformers Excited Above xxx.x% over Tap.
> XSYSTEMLOSS - Transmission System Losses.
> BPALoads - BPA Industrial Loads.
> DCSYSTEM - DC System.
> SHUNTsum - Shunt Reactive Summary.
> SUMLTC - Summary of LTC Transformers. - Summary of LTC Reactive Utilization
> SUMPHASE - Summary of Phase-shifters.
> SUM%VAR - Summary of %Var-controlled buses. - Summary of AGC Control - Summary of Line
↳Drop Compensation
> SUMBX - Summary of Type BX buses. > SUMRAN - Summary of Adjustable Var compensation.
> SERIESCOMP - Transmission Lines Containing Series Compensation.
> BUS - Bus Quantities. > SPIN - Spinning Reserves.
> LINEEFF - Transmission Line Efficiency Analysis. (Lines Loaded Above xxx.x % of
↳Nominal Ratings).
> TRANEFF - Transformer Efficiency Analysis. - Total Losses Above xx.xx % of Nominal
↳Ratings.
> TRANLOSS - Transformer Efficiency Analysis - Core Losses Above xx.xx % of Nominal
↳Ratings.
```

## 2.6.12 BRANCH\_DATA

```
/ BRANCH_DATA, FILE = <filespec>, DATE = <myy>,
      BUSDATA_FILE = *
      <filespec>
```

This command specifies that the base case will be established from a master branch data file and associated bus data file. Branch data selected from this file will have an energization date (date in) and a de-energization date (date out) corresponding with the DATE specified on the above command.

If BUSDATA\_FILE is not specified or has parameter value \*, the program expects bus data to follow in the input stream.

See MERGE\_OLD\_BASE and MERGE\_NEW\_BASE for more information about branch data file merging. Using the MERGE\_OLD\_BASE and MERGE\_NEW\_BASE commands is preferred.

## 2.6.13 BUS\_SENSITIVITIES

```
/ BUS_SENSITIVITIES
```

The primary motive of sensitivity is to calculate the instantaneous system response to sudden capacitor switching operations. This is difficult to model in the Powerflow because all LTCs must be turned off. This may cause solution divergence because LTCs are an integral part of any DC system. This problem is circumvented using sensitivities.

By recalculating the Jacobian matrix, various constraints can be changed. The flexibility of these constraints is evident in the format of the sensitivity command.:

```
/ BUS_SENSITIVITIES, LTC=ON, AI_CONTROL=CON, Q_SHUNT=ADJ, Q_GEN=ADJ
      OFF,                OFF      FIXED      FIXED
      MON
```

The top line defines the default values.

The first two options correspond with the standard solution options. The second two options define the conditions in which type BQ and BG buses can operate holding constant voltage.

For example, enabling the option Q\_SHUNT = FIXED, type BQ buses have all shunt fixed. If there is no rotating machinery ( $Q_{max}$  and  $Q_{min}$  are zero), then the bus holds constant  $Q$  ( $PQ$ ). Since type BG buses always have Q\_shunt fixed, this option has no affect on generator buses.

Similarly, by enabling the option Q\_GEN = FIXED, type BQ and BG buses have all generation fixed and operate in state  $PQ$ . Type BG buses will operate in state  $PQ$ . If BQ buses have no shunt, they also will operate in state  $PQ$ .

In order of time response, the generators respond within several seconds. Thus, Q\_GEN will normally be adjustable. LTC's, DC LTC's, and switched capacitors are controlled by time-delayed voltage relays to minimize spurious operation.

LTC's 0.5 - 3.0 minutes

DC LTC's 5 seconds

CAP/REACTORS:5 - 30 seconds

The slowest component is area interchange control. Its response time is 0.5 to 10 minutes.

By appropriate selection of options, the Jacobian matrix can represent nearly any time frame of response.

## Selected Buses

Following the BUS\_SENSITIVITIES record, individual buses are selected for perturbation. These buses are identified by the B formatted records that follow them. A maximum of 50 buses may be specified.

The perturbed quantity is identified by nonzero entities in one of the fields: P\_load, Q\_load, G\_shunt, B\_shunt, P\_generation or Q\_generation.

The fields on the B-blank record determine which sensitivity  $\frac{dP}{d\theta}$ ,  $\frac{dP}{dV}$ , or  $\frac{dQ}{dV}$  is computed.

B ID	S CH ID	O W N E R	NAME										KV	Z O N E	LOAD				SHUNT		P M A X M W	P G E N M W	Q S C H E D M V A R	Q M I N M V A R	V H O L D V M A X P U	V M I N P U	REMOTE BUS				% S P A R S E									
																			LOAD = + CAP = + REA = -																					
			1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8
B																																								
B																																								
BV																																								

- ☐ = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.  
☐ = Optional; blanks or zeros are acceptable.  
☐ = Not Applicable.

Fig. 2.6.2: B-blank Record Sensitivity Fields

Table 2.6.4: Field Values for Sensitivities

Sensitivity	Column 2	PLOAD, PSHUNT, PGEN	QLOAD, QSHUNT, QSCHED
$\frac{dP}{d\theta}$	Not required	Required	Not required
$\frac{dP}{dV}$	Literal: "V"	Required	Not required
$\frac{dQ}{dV}$	Not required	Not required	Required
$\frac{dQ}{dV}$ *	(blank)	(blank)	(blank)

\* This is the default.

## Repeat Sensitivities

A powerful feature of the sensitivity process is the ability to refactor the Jacobian matrix under different control schemes. For example, one / BUS\_SENSITIVITIES record could enable only the Q\_GEN option (exciters on, everything else off) for an instantaneous response. Following the necessary B formatted records a second / BUS\_SENSITIVITIES record could enable all options for a long term response. Assuming the same bus list is repeated, then a comparison between the two corresponding sensitivities would yield the short-term and long-term effects of the bus's injection perturbation.

### Example

The following is an actual case. Bus OLYMPIA 230 was specified for a -172 MVAR shunt application. If Q\_Load or Q\_Generator was specified, the actual Q\_Perturbation would be -172 MVAR. For Q\_Shunt, the Q\_Perturbation is calculated.:

```
Delta (Q)= Q_Shunt * V**2
= -172 * (1.067)**2
= -195.82 MVAR
New_Voltage (kV) = Old_Voltage (kV) + Sensitivity * Delta_Q
= 245.45 kV + 0.0334 * (-195.82)
= 245.45 kV - 6.54 kV
= 238.91 kV
```

The correct computed value on the listing is 238.90 kV. The different figures in the example are due to round off.

The correlation with actual Powerflow cases is very close. The calculated voltage excursion -6.54 kV is within two percent of the actual excursion. The accuracy is significant because the actual and estimated voltages will differ 0.001 per unit at most!

### Sample Deck Setup

```
(POWERFLOW,...)
.
.
.
/ CHANGES
.
.
.
/ BUS_SENSITIVITIES,LTC=ON,AI_CON=OFF,Q_SHUNT=ADJ,Q_GEN=ADJ
B    MONROE    500    316
```

### Output

```
BUS_SENSITIVITIES COMPUTED WITH THE FOLLOWING CONTROLS:

      LTC CONTROL          ( )  OFF
      ----            (X)  ON (FULL CONTROL      )
                        ( )  ON (NO VOLTAGE CONTROL)

      AI CONTROL           (X)  OFF
      --              ( )  CONTROL (DEFAULT)
                        ( )  MONITOR

      Q_SHUNT CONTROL      (X)  ADJUSTABLE
      -----            ( )  FIXED

      Q_GENERATION CONTROL (X)  ADJUSTABLE
      -----            ( )  FIXED

0 MONROE    500.0 A  316.0 MVAR CHANGE IN SHUNT AT THIS BUS WILL CAUSE THE FOLLOWING
↪ VOLTAGE CHANGES
```

(continues on next page)

(continued from previous page)

0	BUS	BASE	BUS	BASE	----- DVOLT/DQ	-----	----- BASE VOLTAGE
→	-----	----- NEW VOLTAGE	-----		(P.U./P.U.)	(KV/MVAR)	(P.U. KV)
→	(KV)	(P.U.KV)		(KV)			
	MV-SVC	19.6			-0.0106	-0.0023	1.118
→	21.92	1.081		21.10			
	MONROE	500.0			0.0061	0.0321	1.052
→	525.84	1.073		537.05			
	WRK 25	25.2			0.0040	0.0010	1.041
→	26.22	1.054		26.59			
	MONROE	230.0			0.0039	0.0092	1.037
→	238.54	1.051		241.77			
	GDK 4	4.0			0.0039	0.0002	1.049
→	4.20	1.063		4.25			
	MONROE T	230.0			0.0038	0.0090	1.035
→	237.98	1.048		241.12			
	RIM 12	12.6			0.0037	0.0005	1.041
→	13.11	1.054		13.29			
	RYL 12	12.6			0.0037	0.0005	0.996
→	12.55	1.009		12.71			
	WRK 60B2	60.0			0.0037	0.0023	1.020
→	61.17	1.033		61.97			
	WRK 60B1	60.0			0.0037	0.0023	1.020
→	61.22	1.033		62.01			

## 2.6.14 CHANGE\_BUS\_TYPE

/ CHANGE\_BUS\_TYPE

This command disables voltage control in selected areas of the system and performs bus type changes from a voltage control type to a more passive type. The changes it makes are permanent and apply to the case in residence. If this command appears before any system changes, the bus type changes will apply before the system changes, exempting any new or changed buses. If this command appears after any system changes, any new or changed buses will be subject to bus type changes invoked with this command.

An example is shown below.

```
/ CHANGE_BUS_TYPE, BQ=B, BG=BQ, BT=B , BX=B , -
    LTC = OFF, -
    AREAS=<area_1,...>, -
    ZONES=<zone_1,...>, -
    LIST=ON
```

### Bus type changes

Four types of buses may be changed: BQ, BG, BT and BX. All possible bus type transitions are depicted above using the format <old\_type> = <new\_type>.

The full repertoire is listed in the table below. (Note that some restrictions apply.)

Table 2.6.5: Bus Type Changes

Change	Restriction
BQ → B	If PGEN = 0.0 and QGEN =0.0
BQ → BF	If PGEN ≤ 0.0.
BQ → BF*	Unconditional.
BG → BQ	If PGEN > 0.0.
BG → B	If PGEN ≤ 0.0.
BG → BF	If PGEN = 0.0.
BG → BF*	Unconditional.
BT → B	(This deletes any adjacent LTCs which are controlling BT nodes.)
BX → B	
BX → BF	
BX → BF*	

### LTC Transformer Control

A bus type change BT=B will delete LTCs only if the controlled bus is type BT. A more general option:

LTC = OFF

disables all LTCs within the specified area except for any dc commutating transformers.

### LIST

The LIST parameter accepts two values – ON and OFF. The default is ON. This applies to the CHANGE\_BUS\_TYPE summary where the initial and final state of each bus affected is depicted. Setting LIST=OFF is recommended for repetitious batch runs.

### Excluded Buses

Means are available to exempt individual buses from type changes defined in the / CHANGE\_BUS\_TYPE command. These buses are excluded with the following command:

```
>EXCLUDE_BUS
B          name      base
B          name      base
. . .
. . .
```

## Line Drop Compensation

This feature temporarily replaces the ordinary BG -> BC voltage control of a remote bus with a BG control of a compensated voltage, which is specified as a percentage within the step up transformer. This control scheme is valid only for this case, and may be introduced only within context of a `CHANGE_BUS_TYPE` command. In subsequent cases, these generators revert to their normal control mode

The target compensated voltage is defined with a computed voltage limit. That limit is derived from two base case terminal voltages – the BG bus and the remote BC bus (the remote bus may be another type). The formula used is

$$V_{max} = V_{min} = PCT * V_{BG}^{base} + (1 - PCT) * V_{BC}^{base}$$

Example:

```
/ CHANGE_BUS_TYPE, BQ = B, BX = B, BG = BQ
...
> LINE_DROP_COMPENSATION
BG COULEE 13.8, 70%
BG CHIEF J013.8, 80%
```

**Restrictions on Line Drop Compensation** The following restrictions apply to line drop compensation:

- All buses selected for Line Drop Compensation must be type BG. All buses selected are exempt from any bus type change BG -> BQ or BG -> B.
- The controlled remote bus must be immediately adjacent to the generator.
- The specified percentage should be between 0 and 100%. A warning is issued if the specified percentage is outside this range.
- A maximum of 20 generators may be selected for line drop compensation.
- The line drop compensation is case specific. It defines the base solution, but is not saved on the base history data file.

## Reactive Compensation

This feature is similar to Line Drop Compensation; it temporarily replaces the ordinary BG -> BC voltage control of a remote bus with a BG control of a compensated voltage, which is specified as the voltage drop from the bus terminal voltage computed with the generator reactive power in series with a user-specified impedance. This control scheme is valid only for this case, and may be introduced only within context of a `CHANGE_BUS_TYPE` command. In subsequent cases, these generators revert to their normal control mode

The target compensated voltage is defined with a computed voltage limit. That limit is derived from two base case terminal voltages – the BG bus and the remote BC bus (the remote bus may be another type). The formula used is

$$V_{max} = V_{min} = V_{BG}^{base} - \frac{Q_{BG}}{V_{BG}} * X_{BC}$$

$$X_c = \frac{PCT}{100} * \frac{100}{MVA_{base}}$$

where  $PCT$  is the user-specified percentage and  $MVA_{base}$  is the user-specified machine MVA.

Example:

```
/ CHANGE_BUS_TYPE, BQ = B, BX = B, BG = BQ
...
> REACTIVE_COMPENSATION
```

(continues on next page)

(continued from previous page)

```
BG COULEE 13.8, 5%, 100
BG CHIEF J013.8, 5%, 100
```

**Restrictions on Reactive Compensation** The following restrictions apply to reactive compensation are identical to those which apply to line drop compensation:

- All buses selected for Reactive Compensation must be type BG. All buses selected are exempt from any bus type change BG -> BQ or BG => B.
- The controlled remote bus must be immediately adjacent to the generator.
- The specified percentage is typically in the range 5-6%. It may be negative if the voltage is internal to the machine.
- A maximum of 20 generators may be selected for reactive compensation.
- The reactive compensation is case specific. It defines the base solution, but is not saved on the base history data file.

**Output Reports** A special summary of all line drop compensation buses is listed in the analysis group under the title Summary of Line Drop Compensation. It is available either as a level 4 option on the /P\_ANALYSIS or /F\_ANALYSIS command or as the SUM%VAR option on the /ANALYSIS\_SELECT command.

```
/ANALYSIS_SELECT
SUM%VAR
```

### Example

An example is shown below.:

```
( POWERFLOW, ... )
...
...
...
/ CHANGES, ...
...
...
...
/ CHANGE_BUS_TYPE, BG=BQ, BT=B ,BX=B, -
                LTC = OFF, -
                AREAS = NORTHWEST, LIST=OFF
/ CHANGE _BUS_TYPES,BQ=B,BX=B,BG=BQ,LTC=OFF,AREAS=NORTHWEST,BC=HYDRO
>EXCLUDE_BUSES
B   CENTRALA20.0
B   BONN PH213.8
B   BONNVIL213.8
B   DALLES 313.8
B   DALLES2113.8
B   DALLES2213.8
/ SOLUTION
( END )
```

In this example, the disabling of remote voltage control is restricted to area NORTHWEST. Within this area, all BG generators are permanently changed to type BQ; all LTCs are disabled; and all BX buses are frozen to their discrete value.



## 2.6.15 CHANGE\_PARAMETERS

```

/ CHANGE_PARAMETERS, BUS=<name, base>,V=<set_value>,Q=?
                                VX=<set_value>,QY=?
                                Q=<set_value>,V=?
                                QY=<set_value>,QV=?
                                P=<set_value>,V=?
                                P=<set_value>,V=?

> BX = LOCKED
B OSTRNDR 500
B MALIN 500

```

or

```

/ CHANGE_PARAMETERS, BUS=<name, base>,V=?, -
    %LOAD_CHANGE,%PY=<##>, %QX=<##>, -
    %PX=<##>, %QY=<##>, -
    ZONES=NA, NB, ..., -
    OWNERS=BPA, PSP, ..., -
    AREAS=NORTHWEST

```

or

```

/ CHANGE_PARAMETERS, BUS=<name, base>,V=?, -
    %GEN_CHANGE,%PY=<##>, %QX=<##>, -
    %PX=<##>, %QY=<##>, -
    ZONES=NA, NB, ..., -
    OWNERS=BPA, PSP, ..., -
    AREAS=NORTHWEST

```

**Note:** This is one of the three commands which are order-dependent on the /SOLUTION command (the other two commands are LINE\_SENSITIVITIES and LOSS\_SENSITIVITIES). Each of these must follow the /SOLUTION command.

The /CHANGE\_PARAMETERS command perturbs a specified network parameter immediately after a successful solution, and initiates a new solution. The process continues until the last /CHANGE\_PARAMETERS command has been read. All changed network parameters are permanent in the base case in residence. The output, analysis, and saved base case reflecting the final values of the parameters from the last change.

The /CHANGE\_ANALYSIS feature is extremely useful to quickly and accurately generate a set of points for plotting Q-V and P-V curves. When used in conjunction with /USER\_ANALYSIS, the values of additional network quantities can be extracted during each /CHANGE\_PARAMETERS, enriching the scope of examination into the network.

The distribution VX, VY, etc., designates both the quantity and the axis on the X-Y data file. Default values (V, Q, etc.) are shown in Table 4-6.

### BX = LOCKED

Type BX buses selected with this feature emulate the characteristics of mechanically switched shunt capacitors (MSC) controlled by a voltage relay. This voltage relay operates within a voltage deadband ( $V_{min}$ ,  $V_{max}$ ):

1. If  $V_{min} < V < V_{max}$ , then freeze present  $X_{shunt}$  value.
2. If  $V < V_{min}$ , switch in additional capacitor steps or switch out connected reactor steps to raise the voltage, one step at a time.
3. If  $V > V_{max}$ , switch out connected capacitor steps or switch in additional reactor steps to lower the voltage, one step at a time.

For exposition, the feature is called BX Locking. In the absence of this feature, the normal operation is to switch  $X_{shunt}$  one step per iteration to bias the bus voltage to  $V_{max}$ .

### Restrictions on BX Locking

The following restrictions apply to BX locking:

- Only bus type BX buses may be selected for BX locking.
- The feature is limited to a maximum of 10 BX locked buses.
- This feature can be inserted after any / CHANGE\_PARAMETERS command. It defines *when* BX switching on selected BX buses becomes locked. Once defined, BX locking remains in effect for the duration of the study.
- The voltage limits may be temporarily modified for BX locking. The new voltage limits are entered in columns (58:65) in the ordinary manner. These limits are temporary. After the solution, the original limits will be used for analysis reports.
- The BX locking feature is not saved on any generated base case.

### Bus Perturbation

Two forms of / CHANGE\_PARAMETERS are shown. The first form is bus perturbation. Three types of quantities may be perturbed:

V = <set\_value> Perturbs the Bus voltage magnitude (p.u.).

P = <set\_value> Perturbs the  $P_{gen}$  (MW).

Q = <set\_value> Perturbs the  $Q_{gen}$  (MVAR).

### Restrictions

V-perturbations are applied on V-constrained buses: BQ not at Q-limits, BE and BS types. If the bus type is unacceptable, it is automatically changed to a type BE and a warning diagnostic is issued.

Q-perturbations are applied on Q-constrained buses: B, BC, BT and BQ in state Q\_min or Q\_max. If the bus type is unacceptable, it is automatically changed to a type B and a warning diagnostic is issued.

P-perturbation can only be applied on P-constrained buses: all types except BS, BD, BM, and area slack buses.

## Load Perturbation

The second form of / `CHANGE_PARAMETERS` is `LOAD` perturbation. Either the `P_load` or the `Q_load`, or both, may be perturbed a set percentage.

If no `ZONES`, `OWNERS`, or `AREAS` are specified, the percentage change applies to the entire system.

Note that the %P or %Q quantities in the output file correspond to the load that is changed. It may not be the total system load.

The inclusion of `OWNERS` with either `ZONES` or `AREAS` select candidates that are mutually inclusive.

Note that continuation records are accepted here.

For best results, the `%LOAD_CHANGE` option should be used in conjunction with `GEN_DROP`. Otherwise, all increase in load is picked up by the area and system slack buses.

## Generation Perturbation

The third form of / `CHANGE_PARAMETERS` is `GENERATION` perturbation. Either the `P_gen` or the `Q_gen`, or both, may be perturbed a set percentage.

If no `ZONES`, `OWNERS`, or `AREAS` are specified, the percentage change applies to the entire system.

Note that the %P or %Q quantities in the output file correspond to the generation that is changed. It may not be the total system generation.

The inclusion of `OWNERS` with either `ZONES` or `AREAS` select candidates that are mutually inclusive.

Note that continuation records are accepted here.

For best results, the `%GEN_CHANGE` option should be used in conjunction with `GEN_DROP`. Otherwise, all increase in generation is compensated by the area and system slack buses.

## Bus Monitored Quantities

Two types of bus quantities can be monitored:

V = ? Monitors the voltage magnitude.

Q = ? Monitors the reactive allocation including short and unscheduled VARS.

Restrictions apply. V = ? pertains to a BE bus. Q=? pertains to a B bus. Warning diagnostics flag these conditions.

## PLOT File

Each / `CHANGE_PARAMETERS` command generates an x,y plot point in a file with subtype `.QVPT`.

The composition of these points is dependent upon the composition of the `CHANGE_PARAMETERS` command. The table below summarizes the output.

Table 2.6.6: CHANGE\_PARAMETERS Values in .QVPT File

Set Point	Monitored Point	X Value	Y Value
V=<##>	Q=?	Q	V
Q=<##>	V=?	Q	V
P=<##>	Q=?	Q	P
P=<##>	V=?	P	V
%P=<##>	V=?	P_Load	V
%P=<##>	Q=?	P_Load	Q
%Q=<##>	V=?	Q_Load	V
%Q=<##>	Q=?	Q_Load	Q

Each line in the QVPT file is interpreted in the .PFO (power flow output) file.

Example

```
PLOT POINT 3 X (Q) = ##### Y (V) = #####
```

This statement says point  $X_3$  pertains to  $Q$  and  $Y_3$  pertains to  $V$ .

## User Analysis

To circumvent the limitations of monitoring a single bus's  $V$  or  $Q$ , additional quantities may be monitored using a user-defined analysis file defined with the / USER\_ANALYSIS command.

The user analysis file is processed for each encountered / CHANGE\_PARAMETERS command. Its output is appended into an output file with subtype .USR\_REPORT

## Example 1

```
/ CHANGES, FILE= *
.
/ CHANGE_BUS_TYPES, BQ=B, BQ=BQ, BX=B, LTC=OFF, AREA=NORTHWEST
.
/ SOLUTION
.
/ CHANGE_PARAMETERS, BUS = RAVER      500., VY = 1.065, QX = ?
/ CHANGE_PARAMETERS, BUS = RAVER      500., VY = 1.060, QX = ?
/ CHANGE_PARAMETERS, BUS = RAVER      500., VY = 1.055, QX = ?
.
.
.
/ CHANGE_PARAMETERS, BUS = RAVER      500., VY = 1.000, QX = ?
(END)
```

In this example, buses in area NORTHWEST with types BQ, BG, and BX were changed to bring about a freeze in voltage control. The / SOLUTION command is a dummy command, introduced to illustrate the position of the pure / CHANGE\_PARAMETERS commands. If the bus name following the BUS = keyword has imbedded blanks, insert a pound sign (#), for example, BELL#BPA.

At the conclusion of an ordinary successful solution, the / CHANGE\_PARAMETERS records are processed, one by one. The first encounter will internally change the bus type of RAVER 500 to BE, if it is another type, and set its voltage to  $V = 1.065 p.u.$  The perturbation will force a new Newton-Raphson solution. The  $Q$  of RAVER is monitored. Its perturbed solved values will be printed out.

Subsequent / CHANGE\_PARAMETERS commands will perform additional perturbations.

## Example 2

```

/ USER_ANALYSIS, FILE=DRB2:[EOFBJL]USANLINE.DAT
/ CHANGE_BUS_TYPES, BQ=B, BX=B, BG=BQ, LTC=OFF, AREA=NORTHWEST, BC=HYDRO
/ CHANGE, FILE= *
.
.   THIS CASE MODELS THE P-V CURVE FOR THE POST TRANSIENT
.   CONDITIONS FOLLOWING
.   LOSS OF THE COULEE - RAVER #1 500 kV LINE.
.   INSTALL LINE DROP COMPENSATORS ON COULEE
.   500 UNITS AND JOHN DAY
.   AND ALL DALLES UNITS (EXCEPT 115 kV) AND
.   BONNEVILLE (EXCEPT 115 kV)
.   AND CENTRALIA AND CHIEF JOE
.   300 MVAR SVC AT KEELER AND MAPLE VALLEY
.
BGM   CENTRALA20.0
BGM   BONN PH213.8
.
/ GEN_DROP, INIT=75, AREA=NORTHWEST, BC=HYDRO
B      LIBBY      13.8,   PMIN= 289.2,   PMAX=289.2
.
/ SOLUTION
>AI_CONTROL=MON
.
.MONITOR RAVER 500 VOLTAGE AND INCREASE ZONE NA LOAD
.
/ CHANGE_PARAMETERS, BUS = RAVER      500., V = ?
                                %LOAD_CHANGE %P = 0.5, %Q = 0.5, ZONES = NA
/ CHANGE_PARAMETERS, BUS= RAVER      500., V= ?
                                %LOAD_CHANGE %P = 0.5, %Q = 0.5, ZONES = NA
/ CHANGE_PARAMETERS, BUS= RAVER      500., V= ?
                                %LOAD_CHANGE %P = 0.5, %Q = 0.5, ZONES = NA
.
.
(END)

```

### Miscellaneous Notes

If the system is severely perturbed, / CHANGE\_PARAMETERS will cause divergence. If this happens, it is assumed that subsequent perturbations will be severe, so divergence will cause them to be ignored. A diagnostic will be issued.

### 2.6.16 CHANGES

```
/ CHANGES, FILE = file_name
```

This command introduces system data change records. Column 3 on all bus, branch, area interchange and area intertie records contain a change code:

blank Add.

M Modify (non-blank fields are changes).

D Delete.

R Restore (previously deleted, available only on (NEXTCASE) runs).

The following is a list of some specialized change commands.

DA Delete all buses within named areas.

DZ Delete all buses within named zones.

PO, PZ, PN, PA Perform percentage changes according to type.

FILE is optional. If included, records in that file will be processed before any additional change records, which may be in the input stream.

### 2.6.17 COMMENT

```
/ COMMENT
```

This command introduces comment records into the output report. The comments will appear at the beginning of some output listings. The /COMMENT command is optional; all C comments in the bpf control file will be processed.

Comment text must have a C in column 1. Up to 20 comment records are permitted. Comment text is put in columns 2-80. Comments are saved in any NEW\_BASE file for use when getting a plot.

When bpf loads a base file, any previous comments are deleted, then all comments in the bpf control file are added. The result is that only the comments in the bpf control file are saved.

### 2.6.18 COMMON\_MODE\_ANALYSIS

```
/COMMON_MODE_ANALYSIS, BASE_FILE = <base_file_name>,  
COMMON_MODE = <common_mode_file>,  
OUTAGE_FILE = <outage_file>,  
OUT_FILE = <output_file>
```

This command combines features of a common mode file used in the CFLOW program pvcure with the output reports used in the Outage Simulation program, in effect emulating a “slow outage” program. It was written specifically to accept the pvcure input file without modification.

The outages, defined as MODE within the script in the COMMON\_MODE file, typically consists of a sequence of commands /CHANGE\_BUS\_TYPES, /CHANGES, /SOLUTION, and /GEN\_DROP. The mode itself is defined by name on a leading >MODE record; its composition is defined with the change records following a /CHANGES command.

At the end of each >MODE set contained within the file named in the COMMON\_MODE command, the solution results (or divergence state) are analyzed: line overloads and bus under/over voltages are written to the user-specified output file in the same format for the OUTAGE\_SIMULATION program.

The capability to restrict the analyzed output to subsystems defined with base kV's and zones as is now done in the OUTAGE\_SIMULATION program also exists in this feature. That is the purpose of the OUTAGE\_FILE. The OUTAGE\_FILE is a bone fide OUTAGE\_SIMULATION file which processes only two of its commands: >OVERLOAD and >OUTAGE. All others are ignored. (The >OUTAGE command is used only if the >OVERLOAD command is missing and becomes a clone of an implied >OVERLOAD command.)

## Description of Operation

Three phases are involved.

1. Initialization phase. The /COMMON\_MODE\_ANALYSIS record is parsed and the relevant input and output files are opened.
2. The main loop to process >MODE records. The base case in residence is reloaded and the associated processes within the >MODE set are processed exactly in the manner performed in the batch powerflow program. At the conclusion of a solution the output results (line overloads, bus under/over voltages, and any solution divergence) are tabulated in internal arrays.
3. At the conclusion of the last >MODE command, the tabulated results are cross-compiled and outputted exactly in the form as is none in the OUTAGE\_SIMULATION program.

### 2.6.19 F\_ANALYSIS\_RPT

```
/ F_ANALYSIS_RPT , LEVEL =4 , *
      1  ZONES = <list>
      2  OWNERS = <list>
      3
```

This command requests that an analysis report for selected zones or owners be added to the microfiche output file. Note that a separate command [FICHE] must be present in order to save anything on microfiche, regardless of printer and analysis options selected.

When <list> is blank, asterisk, or null, ALL is assumed unless limited by a preceding statement.

The level number determines the analysis summaries to be displayed.

For LEVEL=1, the following summaries are included:

- User-defined analysis (optional).
- Buses with unscheduled reactive.

For LEVEL=2, the following are displayed with summaries for LEVEL=1:

- Total system generations and loads by owner.
- System generations, loads, losses and shunts by zones.
- Undervoltage-overvoltage buses.
- Transmission lines loaded above XX.X% of ratings.
- Transformers loaded above XX.X% of ratings.
- Transformer excited above 5% over tap.
- Transmission system losses.

- BPA industrial loads.
- dc system.
- Shunt reactive summary.
- Summary of LTC transformers.
- Summary of phase-shifters.
- Summary of %Var-controlled buses.
- Summary of type BX buses.
- Summary of adjustable Var compensation.
- Transmission lines containing series compensation.

For LEVEL=3, the following is displayed in addition to the LEVEL=2 output:

- Bus quantities.

For LEVEL=4, the following are displayed in addition to the LEVEL=3 display:

- Spinning reserves.
- Transmission line efficiency analysis. Lines loaded above XX.X% of nominal ratings.
- Transformer efficiency analysis. Total losses above X.XX% of nominal ratings.
- Transformer efficiency analysis. Core losses above X.XX% of nominal ratings.

### Example

```
/ F_ANALYSIS_RPT, LEVEL=4, OWNERS= BPA,PGE,PPL,WPS
/ P_ANALYSIS_RPT, LEVEL=1, ZONES = NA, NB, NC
/ F_ANALYSIS_RPT, LEVEL=4, *
/ P_ANALYSIS_RPT, LEVEL=1, ZONES = *
```

### 2.6.20 F\_INPUT\_LIST

```
/ F_INPUT_LIST, FULL,  ERRORS = NO_LIST
                     NONE      LIST
                     ZONES = <list>
                     ZONES = ALL, FULL or NONE
```

This command lists input data on FICHE. Output can be restricted to individual zones specified in <list> and separated with commas. Note that FULL or NONE may be specified in two forms.

The ERRORS option is set to suppress the input fiche if any fatal (F) errors are encountered. This is the default. It can be overridden by setting ERRORS = LIST.



### 2.6.21 F\_OUTPUT\_LIST

```
/ F_OUTPUT_LIST, FULL, FAILED_SOL = FULL_LIST
      NONE                PARTIAL_LIST
      ZONES = <list>      NO_LIST
      ZONES = ALL, FULL or NONE
```

This command lists output on FICHE. Output can be restricted to individual zones specified in <list> which are separated with commas. Note that FULL or NONE may be specified in two forms.

The FAILED\_SOL option is set to override the output listing if a failed solution occurs. It defaults to a full listing. A PARTIAL\_LIST observes zone lists.

### 2.6.22 GEN\_DROP

```
/ GEN_DROP, ..., INITIAL_DROP=#### ...
```

This feature picks up generation from selected generators to balance generation drop. Generation is dropped in one of two ways:

- By system changes with the amount specified under INITIAL\_DROP.
- By PMIN and PMAX limits on selected generators. (These buses are specified with specially formatted B records which follow.)

Generator dropping emulates the short-term characteristics of a system's response where the generation deficit is automatically picked by other machines. The magnitude is presumed to be proportional to PMAX after the effects of the machine's transients have damped out.

Candidate generators that pickup are those in the area of interest with a spinning reserve (a surplus of  $P_{max}$  over  $P_{gen}$ ). The pickup of an eligible machine "i" is allocated proportionally by the ratio

$$GEN\_PICKUP(i) = P_{max}(i) * (TOTAL\_DROPPED / TOTAL\_PMAX)$$

where  $TOTAL\_DROPPED$  is the sum of dropped MW, and  $TOTAL\_PMAX$  is the sum of all candidate machines with spinning reserve.

Some machines may be driven to their  $P_{max}$  limits during reallocation. In this case, the allocation becomes nonlinear and several iterations may be required. A detailed list of each command follows.:

```
/ GEN_DROP, AI_CONTROL=CON, INITIAL_DROP= ####, TOL=####,
      MON
      OFF
      AREAS=<area_1,...> Optional. Do not use with ZONE.
      ZONES=<zone_1,...> Optional. Do not use with AREA.
```

Note that continuation records are acceptable here.

The individual fields of the GEN\_DROP command follow.

### Area Interchange Control

If generation dropping and allocation occurs over several areas, intertie flows may be substantially affected, and it is recommended to change the area interchange from control to monitor unless the new interchange schedule is known.:

```
AI_CONTROL = CON : Control area interchange.  
              OFF : Turn off area interchange.  
              MON : Monitor area interchange.
```

---

**Note:** One other command also affects area interchange control, the >AI\_CONTROL option on the /SOLUTION record. If this follows the /GEN\_DROP command above, it may overwrite the selected option.

---

### Initial Dropped Generation

This is necessary if the dropped generators are deleted or modified in a change case.

```
INITIAL_DROP = ####
```

The field #### denotes the numerical values in MW.

Initial dropped generation may be specified in an alternate method, called the “computed dropped generation.”

### Tolerance

Generation reallocation continues until the mismatch between generation dropped and generation pickup is less than the tolerance. The default value is 10 MW.

```
TOL = ####
```

The field #### denotes the numerical values in MW.

### Areas or Zones

The generation to be picked up may be either system-wide (the default) or restricted to a set of areas or zones.

```
AREAS = <area_1,...>
```

or

```
ZONES = <zone_1,...>
```

The individual areas are separated with a comma (,). If the area name contains a blank, temporarily replace the blank field with a pound sign (#). Continuation records may be employed for aesthetics.

For example,:

```
AREAS =NORTHWEST, -  
      BC-HYDRO, -  
      IDAHO, -  
      MONTANA, -  
      TRANSALTA, -  
      WKOOTENA
```

## Exclude Buses

Means are available to contract the system or subsystem defined in the /GEN\_DROP command. Individual buses may be excluded from participating in generator pick-up. These buses are selected with the following command:

```
>EXCLUDE_BUS
B name base
B name base
```

## Selected Generators To Be Dropped

The amount of generation is defined as the sum of INITIAL\_DROP plus the computed generation to be dropped. The computed generation drop is the amount of violation of P-limits on all specified buses:

$$P_{MIN} < P_{GEN} < P_{MAX}$$

Obviously, only area and slack buses and AGC candidates permit the P-generation to change. Limits can be placed on these buses by specifying a + or - tolerance, or a P<sub>MIN</sub> and P<sub>MAX</sub> (in MWs). P<sub>MIN</sub> keeps slack buses within a narrow range. The special B records introduce these limits explicitly. This is illustrated with the following example:

```
B MORRO 4 18.0, TOL = 20
B MORRO 4 18.0, PMIN = 147, PMAX = 167
```

If the key words PTOL, P<sub>MIN</sub>, or P<sub>MAX</sub> are omitted, P<sub>MAX</sub> is taken from the P<sub>MAX</sub> field on the original or changed bus data record. Recall that on the bus record there is no corresponding field for P<sub>MIN</sub>. Consequently, P<sub>MIN</sub> = 0.0. At least one B record must be present.

## Example

```
( POWERFLOW, ...)
...
...
/ CHANGES, ...
...
... (changes which drop 2450 MW of generation in the Northwest)
...
/ GEN_DROP, AI_CONTROL=MON, TOL=1.0, INIT=2450
BX MORRO 4 18, PMIN = 147, PMAX = 167
/ SOLUTION
( END )
```

**Note:** MORRO 4 is held between 147 and 167 MWs. Dropping 2450 MWs and picking it up elsewhere will change the generation flows and, quite likely, will alter the system losses. The system slack bus accommodates these changes in losses.

### 2.6.23 HEADER

/ HEADER

This command introduces one or two header records into the pagination. Its text will be repeated on the top of each page in the output report. Each header record begins with an H in column 1. It is used to supply the lines of text that will be printed at the top of every page of an output listing, below the standard header1, which contains the caseid, project, program version, and date. These header records are saved in the base case file, and any previous headers are deleted. This is similar to the /COMMENT command.

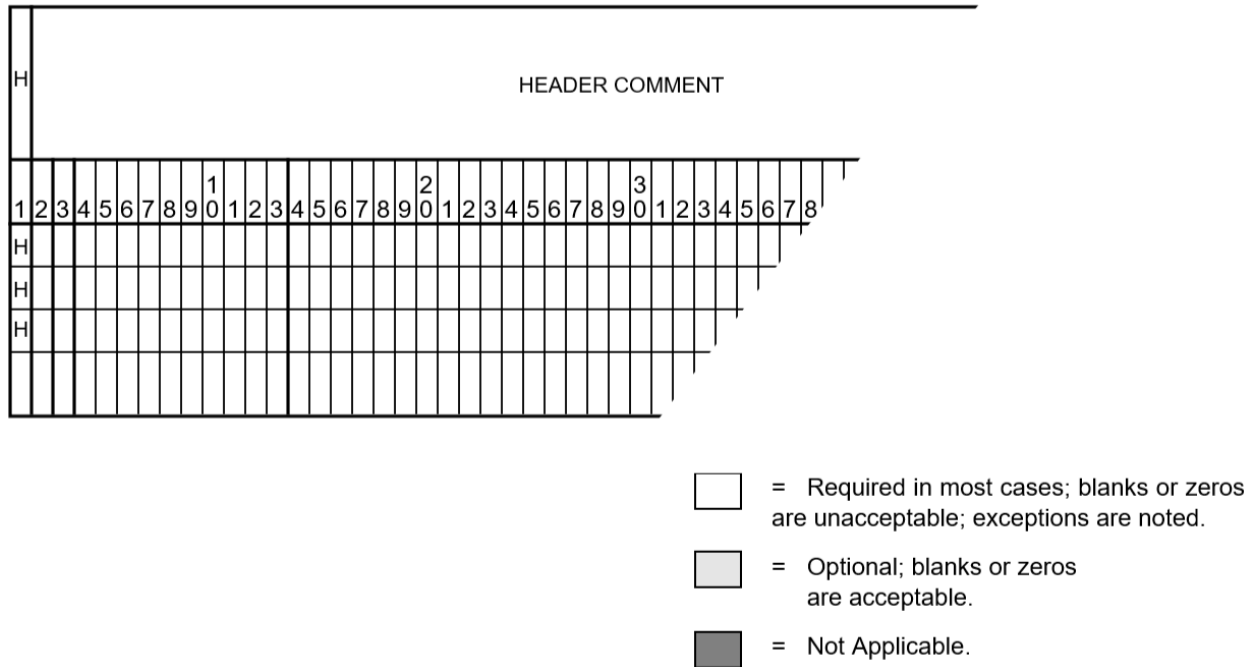


Fig. 2.6.3: Header Comment Input Format

### 2.6.24 INCLUDE\_CONTROL

/ INCLUDE\_CONTROL, file = <filespec>

This command permits the input stream containing commands to be temporarily diverted to the named file. Following an end-of-file, control reverts to the normal input stream. Some restrictions apply. This “included” command file cannot contain any of the following commands:

/ INCLUDE\_CON statement  
 / MERGE\_BASE statement  
 / OUTAGE\_SIM statement  
 / REDUCTION statement  
 / CHANGES statement

### 2.6.25 LINE\_EFF

/ LINE\_EFF, LOADING = <nn>, OWNERS = <list>

Use this command to list lines that are loaded above the prescribed LOADING. The output can be filtered by owners. BPA is the default if no owners are specified.

### 2.6.26 LINE\_SENSITIVITIES

/ LINE\_SENSITIVITIES

**Note:** Three commands are dependent on the SOLUTION command. The commands are CHANGE\_PARAMETERS, LINE\_SENSITIVITIES, and LOSS\_SENSITIVITIES. These three work correctly only if they immediately follow the SOLUTION command.

Line sensitivities relate line immittances (impedance or admittance) to voltage, real power flow, and system losses. Six types are available.

$d \frac{P_{ij}}{dX_t}$  Change in lineflow  $P_{ij}$  with respect to change in transfer reactance  $X_t$ .

$d \frac{P_{ij}}{dB_s}$  Change in lineflow  $P_{ij}$  with respect to change in shunt susceptance  $B_s$ .

$d \frac{Loss}{dX_t}$  Change in system losses with respect to a change in transfer reactance  $X_t$ .

$d \frac{Loss}{dB_s}$  Change in system losses with respect to a change in shunt susceptance  $B_s$ .

$d \frac{V_i}{dX_t}$  Change in bus voltage ( $V_i$ ) with respect to a change in transfer reactance  $X_t$ .

$d \frac{V_i}{dB_s}$  Change in bus voltage ( $V_i$ ) with respect to a change in shunt susceptance  $B_s$ .

The change in transfer reactance  $X_t$  or shunt susceptance  $B_s$  pertains to an existing line. The command statement which invokes line sensitivities is:

/ LINE_SENSITIVITIES, LTC =ON, AI_CONTROL = CON	
OFF	MON
	OFF

The top line depicts default quantities. The options LTC and AI\_CONTROL pertain to LTC transformers and area interchange control.

The second part of the sensitivities is the perturbed quantities  $dX_t$  or  $dB_s$ . They are defined with specially formatted > records and are similar to L records.

Table 2.6.7: Line Sensitivities

Columns	Format	Description
(1:3)	A3	>PB: $\frac{dP_{ij}}{dB_s}$ or $\frac{dP_{ij}}{dX_t}$ >LB: $\frac{dLoss}{dB_s}$ or $\frac{dLoss}{dX_t}$ >VB: $\frac{dV_i}{dB_s}$ or $\frac{dV_i}{dX_t}$
(7:18)	A8,F4.0	Bus1 name and base kV
(20:31)	A8,F4.0	Bus2 name and base kV
(32)	A1	Circuit ID
(33)	I1	Section number
(45:50)	F6.5	Perturbed $X_t$
(57:62)	F6.5	Perturbed $B_s$

A maximum of 50 perturbed quantity > records may be present.

The ambiguity  $d(.) / dB_s$  or  $d(.) / dX_t$  is resolved by non-zero entities for  $X_t$  or  $B_s$ . If both are zero, the default is  $X_t$ . Non-zero entities define the magnitude of the perturbed quantity  $\Delta X_t$  or  $\Delta B_s$ . Perturbed flows, losses, or voltages will be computed using these values.

The perturbed branch flows  $P_{ij}$  are identified with the individual L records that follow. If parallel lines are present,  $P_{ij}$  pertains to the total of all parallel flows.

The perturbed voltages are the 20 largest excursions effected by the change in immittance. The perturbed losses are a simple quantity. An example setup follows:

```
( POWERFLOW, ... )
...
...
...
/ SOLUTION
...
...
...
/ LINE_SENSITIVITIES, AI_CONTROL=ON, LTC=ON
>PB  RAVER      500 TACOMA   500
L    RAVER      500 TACOMA   500
L    GRIZZLY    500 JOHN DAY  500
L    GRIZZLY    500 MALIN    500
L    HANFORD    500 JOHN DAY  500
L    HANFORD    500 LOW MON   500
L    HANFORD    500 OSTRNDER  500
L    HANFORD    500 VANTAGE   500
>LB  RAVER      500 TACOMA   500
>VB  RAVER      500 TACOMA   500
>PB  GRIZZLY    500 JOHN DAY  500
L    RAVER      500 TACOMA   500
L    GRIZZLY    500 JOHN DAY  500
L    GRIZZLY    500 MALIN    500
L    HANFORD    500 JOHN DAY  500
L    HANFORD    500 LOW MON   500
L    HANFORD    500 OSTRNDER  500
L    HANFORD    500 VANTAGE   500
```

(continues on next page)

(continued from previous page)

( STOP )

## Notes

The first perturbation >PB with blank  $X_t$  and  $B_s$  fields requests  $\frac{dP_{ij}}{dX_t}$  (the default). The individual  $P_{ij}$  records.

The second perturbation >LB with blank  $X_t$  and  $B_s$  fields requests  $\frac{dLoss}{dX_t}$  (the default). No L records follow because the monitored quantities are system losses.

The third perturbation >VB with blank  $X_t$  and  $B_s$  fields requests  $\frac{dV_i}{dX_t}$  (the default). No L records follow because the monitored quantities are perturbed voltages. The 20 largest excursions are listed.

## Sample Deck Set-up

```
(POWERFLOW, . . .)
. . .
/ CHANGES
. . .
/ SOLUTION
. . .
/ BUS_SENSITIVITIES, LTC=ON, AI_CON=ON, Q_GEN=ON, Q_SHUNT=ON
B   SATSUP   230   -172
B   OLYMPIA  230   -172
/ BUS_SENSITIVITIES, LTC=OFF, AI_CON=OFF, Q_GEN=ON, Q_SHUNT=FIXED
B   SATSUP   230   -172
B   OLYMPIA  230   -172
(END)
```

## 2.6.27 %LOAD\_DISTRIBUTION

/ %LOAD\_DISTRIBUTION, DISTRIBUTED\_VOLTAGE = NOMINAL\_BASE

This set of commands automatically converts constant power, constant current, or constant impedance loads to a user-specified distribution of constant MVA, constant current, and constant impedance.

The option DISTRIBUTED\_VOLTAGE (or DIST for abbreviated form) selects either NOMINAL (all voltages are 1.0 p.u.) or BASE, which is the individual bus's voltage.

## Constant Current and Impedance Loads

Constant current loads and constant impedance loads are defined by continuation bus (+) records using reserved TYPE s and CODE\_YR s. Constant impedance loads differ from  $G_{shunt}$  and  $B_{shunt}$  quantities in the sense that these quantities are converted into loads and appear in special analysis summaries. The table below describes these special codes and their interpretations.

Table 2.6.8: Special Continuation Bus Types

Type	Owner	CODE_YR	P_LOAD	Q_LOAD	G_SHUNT	B_SHUNT
+A						$B_{xed}$ a
+A		00				$B_{xed}$
+A		01	$P(I)$ b	$Q(I)$ c	$P(G)$ d	$Q(B)$ e
+A		01	$P(I)$	$Q(I)$	$G_{equiv}$ f	$B_{equiv}$ g
+A		02 *I *P	$P(I)$	$Q(I)$	$P(G) P(G) P(G)$	$Q(B) Q(B) Q(B)$

- $B_{xed}$  = Shunt is pi\_back shunt impedance from / CUTTING.
- $P(I)$  = Power is a function of current (constant current).
- $Q(I)$  = Reactive is a function of current (constant current).
- $P(G)$  = Power is a function of shunt  $G$  (constant impedance).
- $Q(B)$  = Reactive is a function of shunt  $B$  (constant impedance).
- $G_{equiv}$  = Shunt is equivalent shunt impedance from / REDUCTION.
- $B_{equiv}$  = Shunt is equivalent shunt impedance from / REDUCTION.

### Description of Constant Current Load Model

For expositional purposes, we will call constant current  $A_{load}$  and  $B_{load}$ . This nomenclature is consistent with the expression for complex current:

$$I = A + jB$$

The power at a constant current load is computed with the expression

$$P_{load} + jQ_{load} = \text{complex}(V) * \text{conjg}(I)$$

where  $\text{complex}(V)$  is the complex voltage and  $\text{conjg}(I)$  is the conjugate of the complex current. The use of the conjugate expression is needlessly complicated for this simple application and has been relaxed. The quantity  $B_{load}$  is stored as its conjugate, that is, no sign reversal is needed to interpret the correct sign of the load in MVAR.

Let  $V$  denote the per unit base or nominal voltage magnitude - depending upon the option DISTRIBUTED\_VOLTAGE. The distributed constant current loads in MW and MVAR are computed as follows:

$$P_{load} = A_{load} * V$$

$$Q_{load} = B_{load} * V$$

Readers may note that this is not true constant current. True constant current loads involve the system phase angle. The modelling here is more lenient: it is constant power factor.

### Description of Distribution Factors

Six percentage distribution factors can be specified by the user. The following example illustrates the relation.

$$P_{load} = 50Q_{load} \text{ From this command, the following quantities will be defined :}$$

$$= 50$$

There are restrictions; the percentage distributions must be complete.

$$PP + PI + PZ = 100.0$$

$$QP + QI + QZ = 100.0$$



This means that if some load is to be unchanged, a value of 100% must be entered for  $P_{load}$  or  $Q_{load}$ .

The following relations hold at the base voltages:

$$\begin{aligned} P_{load_{old}} &= P_{load_{new}} + A_{load} * V + G_{shunt} * V^2 \\ Q_{load_{old}} &= Q_{load_{new}} + B_{load} * V - B_{shunt} * V^2 \end{aligned}$$

where

$$\begin{aligned} P_{load_{new}} \text{ (MW)} &= P_{load_{old}} \text{ (MW)} * PP / 100 \\ Q_{load_{new}} \text{ (MVAR)} &= Q_{load_{old}} \text{ (MVAR)} * QP / 100 \\ A_{load} \text{ (MW)} &= P_{load_{old}} \text{ (MW)} * PI / (100 * V) \\ B_{load} \text{ (MVAR)} &= Q_{load_{old}} \text{ (MVAR)} * QI / (100 * V) \\ G_{shunt} \text{ (MW)} &= P_{load_{old}} \text{ (MW)} * PZ / (100 * V^2) \\ B_{shunt} \text{ (MVAR)} &= - - Q_{load_{old}} \text{ (MVAR)} * QZ / (100 * V^2) \end{aligned}$$

The negative sign for  $B_{shunt}$  is correct. The actual expression is

$$P + jQ = conjg(Y) * V^2$$

A positive value of  $G_{shunt}$  is the same sign as load; a positive value of  $B_{shunt}$  is the same sign as generation.

Those buses whose loads can be distributed can be selected either individually or systematically. Individually selected buses require the > CHANGE\_BUSES command. Systematically selected buses require the > CHANGE\_SYSTEM command.

### Systematically Selected Buses

```
> CHANGE_SYSTEM, PLOAD = ### P + ### I + ### Z,
                   QLOAD = ### Q + ### I + ### Z,
                   AREAS = area_1. ...,
                   ZONES = zones_1, ...,
                   OWNERS = owner_1,
> EXCLUDE_BUSES
B      name      base
B      name      base
B      name      base
```

This example redistributes constant power loads according to the specified percentages.

The redistributed constant current and constant impedance loads are transferred to a new +A01 continuation bus record. The redistributed constant power loads replace the original constant power load.

A special feature has been added to redistribute constant current and constant impedance loads that already have been distributed. As such, these loads are restricted to +A01 and +A02 continuation bus records. The table below describes these options.

Table 2.6.9: Constant Power, Current, and Impedance Keywords

Type of Load to Convert	Keyword for Real Part	Keyword for Reactive Part
Constant Power	PLOAD =	QLOAD =
Constant Current	ALOAD =	BLOAD =
Constant Impedance	RLOAD =	XLOAD =

For example, to change constant current loads, the following commands are used:

```
> CHANGE_SYSTEM, ALOAD = ### P + ### I + ### Z,
      BLOAD = ### Q + ### I + ### Z,
```

The network affected by the specified load change percentages can be restricted to buses within a given subsystem. This subsystem can be defined by those buses having common attributes in two sets:

```
{ AREAS, OWNERS }
```

or:

```
{ ZONES, OWNERS }
```

ZONES and AREAS are mutually exclusive; only one of the above can be specified.

If no owners are specified, all ownerships are implied. The selected subsystem can be further defined by excluding specific bases with the >EXCLUDE option.

More than one set of CHANGE\_SYSTEM commands is permitted. This would permit buses in different areas or zones to have different percentage distribution factors. In case of overlap, precedence is given to the first definition.

### Individually Selected Buses

```
> CHANGE_BUSES, CHANGE_TYPE = PLOAD
      ALOAD
      RLOAD
B   ownname   base   #####   #####   #####   #####   #####   #####   PLOAD
B   ownname   base   #####   #####   #####   #####   #####   #####   ALOAD
+x  ownname   base yr #####   #####   #####   #####   #####   #####
+x  ownname   base yr #####   #####   #####   #####   #####   #####
```

This command permits unique distribution factors to be specified for individual buses. The buses and their distribution factors are identified on fixed field records. The format of the B % load change record is shown in the figure below. CHANGE\_TYPE is optional. ALOAD and RLOAD have the same interpretation given in Table 4-8. Thus, they would apply to + records, but not to B records.

If the ownership field is blank or includes the bus ownership, the percentages apply only to data on the bus B record. Continuation bus data will not be affected.

On the other hand, if the ownership is the magic code ###, the percentages apply to data on the bus B record and also to data on all associated continuation bus records.

If separate % changes are to apply to bus and continuation bus records, separate + % change records must be used—one for the bus B record and others for the specific + bus records.

The identification fields for + % bus records are identical to those for the + records as in the table below.

Table 2.6.10: Identification Fields for +% Records

Column	Quantity
1	•
2	Type
3- 6	Ownership
7-18	Bus Name and Base kV
19-20	Code Year

B		OWNER	NAME	BASE	Percentage Distribution																								Change Type		
					Loads %						Current %						Impedance %														
					MW			MVAR			MW			MVAR			MW			MVAR											
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2
B																															
B																															
B																															

- ☐ =Required in most cases; blanks or zeros are unacceptable; exceptions are noted.  
☐ =Optional; blanks or zeros are acceptable.

Fig. 2.6.4: CHANGE\_BUS % Load Input Format for B Records

The format of the + % load change records is shown below.

B TYPE	OWNER	NAME	BASE	CODE YR	Percentage Distribution																								Change Type												
					Loads %								Current %								Impedance %																				
					MW				MVAR				MW				MVAR				MW				MVAR																
					1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4		5	6	7	8	9	0	1	2				
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2
+																																									
+																																									
+																																									

- ☐ =Required in most cases; blanks or zeros are unacceptable; exceptions are noted.  
☐ =Optional; blanks or zeros are acceptable.

Fig. 2.6.5: CHANGE\_BUS % Load Input Format for + Records

Wildcards are permitted in these fields: TYPE, OWNER, and CODE\_YR. The wild card character(s) for those fields are #, ###, and ## respectively.

The percentages apply to the distribution of QLOAD and PLOAD to ALOAD and BLOAD, or to RLOAD and XLOAD. They must total 100% each.

If an overlap occurs with the CHANGE\_SYSTEM command, precedence is given to the individually specified buses.

A maximum of 2,000 buses may be specified. Once identified, that bus will be unaffected by any subsequent commands.

### Example 1

PLOAD, QLOAD distributions applied to a bus record. See figure below.:

$$\begin{aligned} \text{PLOAD} &= \% \text{ PL} + \% \text{ PI} + \% \text{ PZ} \\ \text{QLOAD} &= \% \text{ QL} + \% \text{ QI} + \% \text{ QZ} \end{aligned}$$

SUB BT IDEE	SCH BC DEE	OWN R	NAME										kV	ZONE	LOAD				SHUNT		P MAX	P GEN MW	Q SCHED Q MAX MVAR	Q MIN MVAR	V HOLD V MAX PU	V MIN PU	REMOTE BUS										% VARS SD
																			LOAD =+	CAP=+ REA=-							NAME		kV								
																			P MW	Q MVAR							MW	MVAR									
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0								
B			PLOAD												QLOAD																						
B																																					
B																																					

Fig. 2.6.6: Original B Record

1. Remove PLOAD and QLOAD quantities from bus record above.
2. Calculate the following quantities

$$\begin{aligned} \text{P1} &= \text{PLOAD} * \% \text{ PL} \\ \text{Q1} &= \text{QLOAD} * \% \text{ QL} \\ \text{P2} &= \text{PLOAD} * \% \text{ PI} / \text{VOLTAGE} \\ \text{Q2} &= \text{QLOAD} * \% \text{ QI} / \text{VOLTAGE} \\ \text{P3} &= \text{PLOAD} * \% \text{ PZ} / \text{VOLTAGE} ** 2 \\ \text{Q3} &= \text{QLOAD} * \% \text{ QZ} / \text{VOLTAGE} ** 2 \end{aligned}$$

3. Replace the load and shunt fields on the B record and on a new +A\*I record.

Modified B record and new +A\*I record. See below.

B T IDEE	SCH BC DEE	OWN R	NAME	kV	ZONE	LOAD		SHUNT		P MAX	P GEN MW	Q SCHED Q MAX MVAR	Q MIN MVAR	V HOLD V MAX PU	V MIN PU	REMOTE BUS		% VARS SD	
						P MW	Q MVAR	LOAD =+	CAP=+ REA=-										
																MW	MVAR		
																			NAME
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
B																			
+A																			

Fig. 2.6.7: Modified B Record



## Limitations, Restrictions, and Assumptions

The load distribution is presumed to apply to a solved base case. At the base solution, the total load in MWs and MVARs is unchanged after distribution. If the system is not otherwise changed, the solution should converge to the base solution.

Each nonzero load on a bus or continuation bus record generates an associated constant current and constant impedance load on an equivalent +A\*I continuation bus record. The continuation bus array is currently dimensioned for 3360 records.

The number of generated +\*I and +\*P records in a typical base case averages 400 (assuming one for each continuation bus) plus one for each number of nonzero load on the bus records.

BPA's Transient Stability Program in its present form cannot accommodate the Powerflow model of constant current loads.

### 2.6.28 LOAD\_GE

/LOAD\_GE, FILE=<file\_name>

This command imports an ASCII coded GE-formatted network data file

#### LOAD\_GE qualifiers

REFFILE=<reference\_file\_name>

This qualifier introduces a reference base case from which to derive missing ownership and mileage information. The GE data set is potentially richer in content than IPF's base data file, but maybe incomplete if some optional data fields such as mileage or ownerships are omitted.

VERSION=<nnn>

This qualifier defines the version number of the input data. At present, only version 21 is recognized.

RATINGS=(TX=AABC, LN=AAC)

This option correlates the various GE branch ratings with the IPF branch ratings. Four GE transformer ratings (RATEA, RATEB, RATEC, RATED) can be assigned independently to the IPF transformer ratings in the following order – Nominal, Thermal, Emergent, and Bottleneck. Similarly, three GE line ratings (RATEA, RATEB, RATEC) can be assigned independently to the IPF line ratings in the following order – Nominal, Thermal, and Bottleneck.

The default branch IPF ratings, shown in the example above, are assigned per the table below.

This option defines the remotely controlled bus' voltage assignments in the form of bus type and scheduled voltage. The table below describes all options

Table 2.6.11: Remotely controlled bus assigned voltages

Type	Conditional bus type	Action taken
HIGH		$v_{max} = v_{max\_ge}$ $v_{min} = v_{max\_ge}$ Bb -> BT
AVER-AGE		$v_{max} = 0.5 * (v_{max\_ge} + v_{min\_ge})$ $v_{min} = 0.5 * (v_{max\_ge} + v_{min\_ge})$ Bb -> BT
LOW	BQ	$v_{max} = 0.5 * (v_{max\_ge} + v_{min\_ge})$ $v_{min} = 0.5 * (v_{max\_ge} + v_{min\_ge})$ T
LOW	All other types	$v_{max} = v_{max\_ge}$ $v_{min} = v_{min\_ge}$
RANGE	Bb	$v_{max} = v_{max\_ge}$ $v_{min} = v_{min\_ge}$
RANGE	All other types	$v_{max} = 0.5 * (v_{max\_ge} + v_{min\_ge})$ $v_{min} = 0.5 * (v_{max\_ge} + v_{min\_ge})$

## 2.6.29 LOAD\_PTI

```
/LOAD_GE, FILE=<file_name>
```

This command imports an ASCII coded PTI-formatted network data file

### LOAD\_PTI qualifiers

```
REFFILE=<reference_file_name>
```

This qualifier introduces a reference base case from which to derive missing ownership and mileage information.

```
VERSION=<nnn>
```

This qualifier defines the version number of the input data. At present, only version 3 and 4 are recognized.

```
RATINGS=(TX=AABC, LN=AAC)
```

This option correlates the various PTI branch ratings with the IPF branch ratings. Four PTI transformer ratings (RATEA, RATEB, RATEC, RATED) can be assigned independently to the IPF transformer ratings in the following order – Nominal, Thermal, Emergenty, and Bottleneck, Simarily, three PTI line ratings (RATEA, RATEB, RATEC) can be assigned independently to the IPF line ratings in the following order – Nominal, Thermal, and Bottleneck.

The default branch IPF ratings, shown in the example above, are assigned per the table below.

This option defines the remotely controlled bus' voltage assignments in the form of bus type and scheduled voltage. The table below describes all options.

Table 2.6.12: Remotely controlled bus assigned voltages

Type	Conditional bus type	Action taken
HIGH		$v_{max} = v_{max\_pti}$ $v_{min} = v_{max\_pti}$ Bb -> BT
AVER-AGE		$v_{max} = 0.5 * (v_{max\_pti} + v_{min\_pti})$ $v_{min} = 0.5 * (v_{max\_pti} + v_{min\_pti})$ Bb -> BT
LOW		$v_{max} = v_{min\_pti}$ $v_{min} = v_{min\_pti}$ Bb -> BT
RANGE	Bb	$v_{max} = 0.5 * (v_{max\_pti} + v_{min\_pti})$ $v_{min} = 0.5 * (v_{max\_pti} + v_{min\_pti})$
RANGE	All other types	$v_{max} = 0.5 * (v_{max\_pti} + v_{min\_pti})$ $v_{min} = 0.5 * (v_{max\_pti} + v_{min\_pti})$

## 2.6.30 LOSS\_SENSITIVITIES

```
/ LOSS_SENSITIVITIES
```

**Note:** Three commands are dependent on the SOLUTION command. The commands are CHANGE\_PARAMETERS, LINE\_SENSITIVITIES, and LOSS\_SENSITIVITIES. These three work correctly only if they immediately follow the SOLUTION command.

This feature provides valuable information concerning system losses with respect to scheduled active and reactive generation or loads, and to scheduled voltages. The command statement that invokes loss sensitivities is:

```
/ LOSS_SENSITIVITIES, LTC=ON, AI_CONTROL=CON, Q_SHUNT=ADJ,
                        OFF, OFF FIXED
                        QGEN=ADJ,
                        FIXED
```

(continues on next page)

(continued from previous page)

```
AREAS=<area1,area2,...>,
ZONES=<zone1,...>
```

The top line depicts default quantities. The options LTC, AI\_CONTROL, and Q\_SHUNT pertain to LTC transformers, area interchange control, and  $B_{shunt}$  on type BQ buses.

Three loss sensitivities are computed:  $\frac{dLoss}{dP_i}$ ,  $\frac{dLoss}{dQ_i}$ , and  $\frac{dLoss}{dV_i}$ .

These sensitivity computations are linearized about the solved case. For small changes, the sensitivities are extremely accurate. For larger changes, non-linearities redefine the problem. A rule of thumb is that the sensitivities are sufficiently accurate for a 0.5 per unit (p.u.) change in  $P_i$  or  $Q_i$ , and a 0.01 p.u. change in  $V_i$ .

Each sensitivity relates changes in the system losses to a hypothetical change of 1.0 p.u. in scheduled active generation  $P_i$ , reactive generation  $Q_i$ , or voltage  $V_i$ .

Ordinarily, a decrease in system losses is anticipated when  $P_i$ ,  $Q_i$ , or  $V_i$  increases, that is, a negative loss sensitivity.

$$\frac{dLoss}{dP_i}$$

An exception often occurs for  $\frac{dLoss}{dP_i}$ . Occasionally,  $\frac{dLoss}{dP_i} > 0$ , that is, increasing the generation  $P_i$  increases the losses!

Recall the constraint for Area\_i:

$$\text{Area export} = \text{Area generation} - \text{Area load} - \text{Area losses}$$

(Any active bus shunt  $G$  is presumed to be accounted for in area losses.)

Within each area, the generation on the slack bus is adjustable and on all other generators is fixed. Thus, a change of 1.0 p.u. on generator “i” causes two changes in the area slack bus:

- An immediate transfer of -1.0 p.u. to balance the change in generation.
- An additional change to reflect the change in system losses, which are affected by the 1.0 p.u. generation transfer.

Note that the system slack bus or area interchange slack bus must pick up any deficit generation needed to supply loads and system losses. Thus, the sensitivity reflects the change in losses if 1.0 p.u. MW of generation is moved from bus “i” to the system or area slack bus. If the system or area slack bus is closer to the load center, the losses will decrease with the reallocation. Consequently,  $\frac{dLoss}{dP_i} < 0$ . Otherwise, the losses will increase.

$$\frac{dLoss}{dQ_i}$$

A change in reactive generation is quite different from a change in active generation. Changes in reactive generation strongly affect the voltage profile of the system adjacent to bus “i”. Thus, a change in losses is due primarily to the change in voltage profile.



$$\frac{dLoss}{dV_i}$$

A change in scheduled voltage for types BE, BS, BQ, or BG buses directly affects the voltage profile of the system adjacent to bus “i”. Thus, the change in voltages directly affects system losses. In general, higher voltages are accompanied with lower branch currents and hence, lower line losses. Exceptions may occur in cables where large amounts of inductive shunt are necessary to compensate for the capacitance in cable.

### 2.6.31 MERGE\_OLD\_BASE and MERGE\_NEW\_BASE

/ MERGE\_OLD\_BASE and / MERGE\_NEW\_BASE

These subprocesses extract a subsystem from an old base file and merge it with another subsystem to generate a new system. The subsystems are defined by various qualifiers following the MERGE command.:

```
/ MERGE_OLD_BASE,SUBSYSTEM_ID= <subsystem_label>,
  OLDBASE_FILE = <file_spec>
```

and:

```
/ MERGE_NEW_BASE,SUBSYSTEM_ID = <subsystem_label>,
  BRANCH_DATA_FILE = <file_spec>, DATE = <myy>,
  R = <n>, BUS_DATA_FILE = <file_spec>
```

where:

- file\_spec is the file specification for the pertinent file. If file\_spec has the value \* for either the BUS\_DATA\_FILE or BRANCH\_DATA\_FILE, the data is presumed to be the Powerflow command file.
- subsystem\_label is the identifying label for the merged subsystem.
- DATE is the branch extraction date. Branches selected will have their energization date on or before DATE and a de-energization date after DATE.
- The month field (as a digit) also defines winter or summer extended ratings:

```
`m = 1` selects winter peak ratings.
`m = 8` selects summer peak ratings.
```

See Table 4-11 for the complete listing.

- For other values, it is necessary to use an additional parameter R defined in the next section.
- R specifies extended ratings from the branch data file. See Table 4-15 and Table 4-16. Also, see Figure 4-11 and Figure 4-12. Two modes of operation are available:

**\*\* Merge a subsystem from one OLD\_BASE file with another subsystem from a different old base file. \*\***  
Merge a subsystem from an OLD\_BASE file with another subsystem which is newly created from bus and branch records.

The two merge control cards distinguish the source of the subsystem data. / MERGE\_OLD\_BASE identifies an OLD\_BASE file, and / MERGE\_NEW\_BASE identifies the bus and branch records files from which a new subsystem will be constructed.

The R code indicates which extended ratings from the branch data file should be used. For example, the R=2 code in the following card indicates that extra heavy ratings should be used.

```
/MERGE_NEW_BASE,SUBSYSID=BR_BUS,BRAN=BDCY89.DAT,DATE=196,R=2,BUSD=J96EH.BUS
```

Powerflow uses appropriate ratings from the branch data for the peak winter (R=1), peak summer (R=8), extra heavy (R=2), moderate cold (R=3), and spring (R=4) choices.

If the thermal or bottleneck rating on a branch is blank or zero in the columns for extra heavy or moderate cold ratings, the peak winter rating is used (if available). Similarly, if the thermal, bottleneck, or emergency rating on a branch is blank or zero in the columns for spring ratings, the peak summer rating is used (if available).

The chosen ratings are moved to columns 81 through 92 on the branch record in Powerflow.

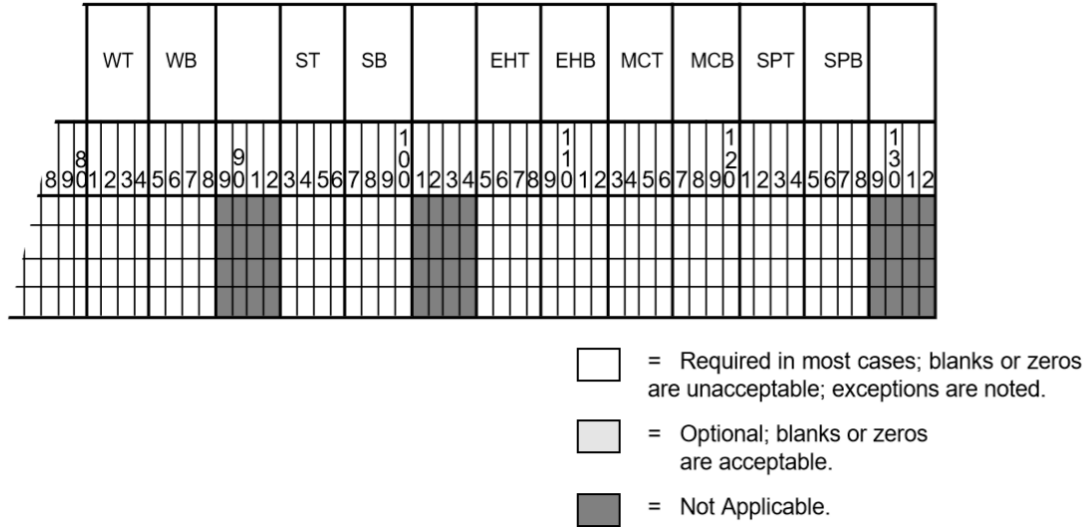





Fig. 2.6.10: Extended Ratings Fields for L and E Records

Table 2.6.13: L and E Record Extended Fields Column Descriptions

Column	Rating ("R" Selection)	Field Description
81-84	1	Winter Thermal (WT)
85-88	1	Winter Bottleneck (WB)
89-92		(not used)
93-96	8	Summer Thermal (ST)
97-100	8	Summer Bottleneck (SB)
101-104		(not used)
105-108	2	Extra Heavy Thermal (EHT)
109-112	2	Extra Heavy Bottleneck (EHB)
113-116	3	Moderate Cold Thermal (MCT)
117-120	3	Moderate Cold Bottleneck (MCB)
121-124	4	Spring Thermal (SPT)
125-128	4	Spring Bottleneck (SPB)

 = Required in most cases; blanks or zeros are unacceptable; exceptions are noted.

 = Optional; blanks or zeros are acceptable.

 = Not Applicable.

This command assigns preference weights on competing interface branches listed following the statement. Two subsystems to be merged are usually topologically complementary and have common branches. These branches are called interface branches. During merging, two sets of competing interface branches vie for selection in the final system. In the absence of any information supplied, the default decision is to select the interface branch with the most detail.

The command above allows the user to assign preferences for the interface branches for each system. Each such interface branch is identified on a standard L, T, or E formatted record. ACCEPT and REJECT must complement each other, or both sets of interface branches will be accepted or rejected.

COMP forces comparison of common interface branches from the two subsystems. Acceptance from one subsystem and rejection from the other is determined on the basis of matching bus and branch ownerships. The assumption is that the bus owner always has better branch data. For BPA users, WSCC data is accepted when branch ownerships cannot be determined from data.:

```
>MERGE_RPT = SORTED
              UNSORTED
```

This command requests the specific level of merge report. This feature is not yet implemented.

```
>RENAME_BUS
```

This command provides a convenient way to resolve potential conflicts of identically named but topologically distinct buses. This command introduces B-formatted records with the old name in columns 7-18 and the new name in columns 20-31.

```
>SAVE_AREAS
```

Use this command to save areas of the subsystem listed following this statement. Each area is identified with a separate A-formatted area record.

```
>SAVE_BASES <list>
```

This command saves buses whose base kV's match the list. Elements of the list are separated by commas (,) and terminated by a period (.).

```
>SAVE_BUSES
```

This command saves listed buses of the subsystem. Saved buses are named on separate B-formatted bus records following.

```
>SAVE_ZONES <list>
```

This command saves listed zones of the subsystem. Elements of the list are separated with commas. Example: >SAVE\_ZONES NA, NB

```
>USE_AIC
```

This command specifies that A records should be generated from the old base file defined by the OLD\_BASE statement.

### 2.6.32 MVA\_BASE

```
/ MVA_BASE = 100
              <number>
```

This command changes the base MVA from the default value of 100 MVA to an assigned value.

### 2.6.33 NETWORK\_DATA

```
/ NETWORK_DATA ,FILE= *,          RXCHECK = ON
                        <filespec>,      OFF
```

This introduces network bus and branch data into the program. No old base case is in residence. `RXCHECK = ON` enables  $R/X$  ratios checking. If the `FILE` parameter value is asterisk (\*), then bus and branch data is assumed to immediately follow this command.

### 2.6.34 NEW\_BASE

```
/ NEW_BASE , FILE = <filespec>
```

This command defines the name of the new base file to save the network solved by the case run. It may be the same as the old base file, if you want to overwrite it.

### 2.6.35 OI\_LIST

This command is used to list ownership interchange.:

```
/ OI_LIST = NONE
           TIELINE
           MATRIX
           FULL
```

Owners are listed using the expanded owner identifications hard-coded in the program. See below for the complete list of codes and expanded names.

Ownership Code	Ownership Name
AAC	ANACONDA ALUMINUM COMPANY
AEC	ATOMIC ENERGY COMMISSION
AEP	ARIZONA ELECTRIC POWER COOPERATION
ALA	ALABAMA POWER COMPANY
ALC	ALUMINUM COMPANY OF AMERICA
APS	ARIZONA PUBLIC SERVICE COMPANY
ARL	ARKANSAS POWER AND LIGHT COMPANY
ARR	ARROWHEAD ELECTRIC COOPERATIVE, INC.
AVC	AMARGOSA VALLEY COOPERATIVE INC.
BBE	BIG BEND ELECT. COOP
BCH	BRITISH COLUMBIA HYDRO AND POWER AUTHORITY
BEC	BASIN ELECTRIC POWER COOP.
BEP	BASIN ELECTRIC COOPERATIVE
BHP	BLACK HILLS POWER AND LIGHT COMPANY
BPA	BONNEVILLE POWER ADMINISTRATION
BPD	BENTON CO. PUD
BRE	BENTON REA
CAL	CALIFORNIA DEPARTMENT OF WATER RESOURCES
CCC	COOS CURRY ELECTRIC COOP
CCP	COWLITZ COUNTY PUBLIC UTILITY DISTRICT NO.1
CCS	CITY OF COLORADO SPRINGS
CE1	DEP. OF ARMY CORPS OF ENGINEER (REGION 1 AREA)

continues on next page

Table 2.6.15 – continued from previous page

Ownership Code	Ownership Name
CE2	DEP. OF ARMY CORPS OF ENGINEER (REGION 2 AREA)
CE3	DEP. OF ARMY CORPS OF ENGINEER (REGION 3 AREA)
CE4	DEP. OF ARMY CORPS OF ENGINEER (REGION 4 AREA)
CE5	DEP. OF ARMY CORPS OF ENGINEER (REGION 5 AREA)
CE6	DEP. OF ARMY CORPS OF ENGINEER (REGION 6 AREA)
CE7	DEP. OF ARMY CORPS OF ENGINEER (REGION 7 AREA)
CEC	COMMONWEALTH EDISON COMPANY (ILLINOIS)
CED	COMMONWEALTH EDISON COMPANY OF INDIANA, INC.
CEN	CENTRAL POWER ELECTRIC COOP., INC. (N. DAKOTA)
CIP	CENTRAL IOWA POWER COOPERATIVE
CIS	CENTRAL ILLINOIS PUBLIC SERVICE COMPANY
CLA	CLALLAM PUD
CLK	CLARK COUNTY PUBLIC UTILITY DISTRICT NO. 1
CLP	CENTRAL LINCOLN PUD
CLT	CLATSKANIE PUD
CMS	CHICAGO, MILWAUKEE, ST.PAUL AND PACIFIC R.R. CO.
CNP	CENTRAL NEBRASKA PUBLIC POWER AND IRRIGATION DIST.
COB	CORN BELT POWER COOPERATIVE, INC.
COE	DEP. OF ARMY CORPS OF ENGINEERS
COR	CITY OF RICHLAND
CPA	COOPERATIVE POWER ADMINISTRATION
CPD	CHELAN COUNTY PUBLIC UTILITY DISTRICT NO. 1
CPI	CONSUMERS POWER INC.
CPL	CALGARY POWER LIMITED
CPN	C P NATIONAL
CPP	CONSUMERS PUBLIC POWER DISTRICT (NEBRASKA)
CPS	COMMUNITY PUBLIC SERVICE CO.
CPU	CALIFORNIA PACIFIC UTILITIES COMPANY
CRP	COLUMBIA RIVER PUD
CU	COLORADO-UTE ELECTRIC ASSOCIATION
DPC	DAIRYLAND POWER COOPERATIVE (WISC., MINN.)
DPD	DOUGLAS COUNTY PUBLIC UTILITIES DISTRICT
EEQ	EASTERN EQUIVALENT
EIL	EASTERN IOWA LIGHT AND POWER COOPERATIVE
ELE	EL PASO ELECTRIC COMPANY
ELP	EL PASO ELECTRIC COMPANY
EMP	EMERALD PUD
EPE	EL PASO ELECTRIC
ERP	EAST RIVER ELECTRIC POWER COOP.,INC.(S. DAKOTA)
EWE	EUGENE WATER AND ELECTRIC BOARD (OREGON)
FRC	FALL RIVER ELEC. COOP
FRK	FRANKLIN CO. PUD
GH	GRAYS HARBOR COUNTY PUBLIC UTILTIY DISTRICT
GPD	GRANT COUNTY PUD NO.2 (WASHINGTON)
GSU	GULF STATE UTILITIES COMPANY (TEXAS, LOUISIANA)
HAR	HARNEY ELECTRIC COOP
HEA	HIGHLINE ELECTRIC ASSOCIATION
HPL	HOUSTON POWER AND LIGHT COMPANY
IDP	IDAHO POWER COMPANY

continues on next page

Table 2.6.15 – continued from previous page

Ownership Code	Ownership Name
IEL	IOWA ELECTRIC LIGHT AND POWER
IGE	IOWA ILLINOIS GAS AND ELECTRIC COMPANY
IID	IMPERIAL IRRIGATION DISTRICT (CALIFORNIA)
IIG	IOWA-ILLINOIS GAS & ELECTRIC CO.
ILL	ILLINOIS POWER COMPANY
ILM	ILLINOIS AND EASTERN MISSOURI
IME	INDIANA AND MICHIGAN ELECTRIC COMPANY
INL	INLAND POWER AND LIGHT
INP	INLAND POWER AND LIGHT COMPANY
IPC	IDAHO POWER COMPANY
IPL	IOWA POWER AND LIGHT COMPANY
IPS	IOWA PUBLIC SERVICE COMPANY
IPU	IOWA SOUTHERN UTILITIES CO.
ISP	INTERSTATE POWER COMPANY
ISU	IOWA SOUTHERN UTILITIES COMPANY
KCP	KANSAS CITY POWER AND LIGHT COMPANY
KGE	KANSAS GAS AND ELECTRIC COMPANY
KPL	KANSAS POWER AND LIGHT COMPANY
LA	CITY OF LOS ANGELES DEPARTMENT OF WATER AND POWER
LCR	LOWER COLORADO REGION WESTERN AREA POWER ADMIN.
LEC	LANE CO. ELEC.COOP.
LES	LINCOLN ELECTRIC SYSTEM
LEW	LEWIS CO. PUD.
LPL	LOUISIANA POWER AND LIGHT COMPANY
LSD	LAKE SUPERIOR DISTRICT POWER COMPANY
LVP	LOWER VALLEY POWER AND LIGHT
MAI	MAIN-MID-AMERICA INTERPOOL NETWORK
MCM	MCMINNVILLE, CITY OF
MDU	MONTANA-DAKOTA UTILITIES COMPANY
MFR	MILTON-FREEWATER
MH	MANITOVA HYDRO ELECTRIC BOARD
MIN	MINNKOTA POWER COOPERATIVE, INC.
MLC	MISSOURI POWER AND LIGHT COMPANY
MLE	MOON LAKE ELECTRIC ASSOCIATION, INC.
MN1	MASON COUNTY PUD #1
MN3	MASON COUNTY PUD #3
MPC	MONTANA POWER COMPANY
MPL	MINNESOTA POWER AND LIGHT COMPANY
MPO	MISSISSIPPI POWER AND LIGHT COMPANY
MPR	MID PACIFIC REGION - USBR
MPS	MISSOURI PUBLIC SERVICE COMPANY
MPW	MUSCATINE POWER AND WATER
MWD	METROPOLITAN WATER DISTRICT OF SOUTHERN CALIFORNIA
NEP	N.W. ELECTRIC POWER COOP., INC. (MISSOURI,ARK.)
NGT	NEBRASKA ELECTRIC GENERATING AND TRANSMISSION COOP
NIP	NORTHWEST IOWA POWER COOPERATIVE
NLI	NORTHERN LIGHTS, INC.
NPC	NEVADA POWER COMPANY
NPP	NEBRASKA PUBLIC POWER SYSTEM

continues on next page

Table 2.6.15 – continued from previous page

Ownership Code	Ownership Name
NPR	NORTH PACIFIC REGION - USBR
NSC	NORTHERN STATES POWER COMPANY, (WISCONSIN)
NSP	NORTHERN STATES POWER COMPANY, (MINN.,N.D.,S.D.)
NWA	NORTHERN WASCO PUD
NWP	NORTHWESTERN PUBLIC SERVICE COMPANY (S. DAKOTA)
OGE	OKLAHOMA GAS AND ELECTRIC COMPANY
OKP	OKANOGAN CO. PUD
OPD	OMAHA PUBLIC POWER DISTRICT
OPP	OMAHA PUBLIC POWER DISTRICT
OTC	OREGON TRAIL COOP
OTP	OTTER TAIL POWER COMPANY
OWI	OROVILLE-WYANDOTTE IRRIGATION DISTRICT, (CALIF.)
PAN	PORT ANGELES
PDO	PEND OREILLE PUD
PEG	PLAINS ELECTRIC G AND T COOP(NEW MEXICO)
PEN	PENINSULA LT. CO.
PG	PORTLAND GENERAL ELECTRIC COMPANY
PGE	PACIFIC GAS AND ELECTRIC COMPANY
PGT	PLAINS ELECTRIC G AND T COOP., (NEW MEXICO)
PNM	PUBLIC SERVICE COMPANY OF NEW MEXICO
PPL	PACIFIC POWER AND LIGHT COMPANY
PPW	PACIFIC POWER AND LIGHT - WYOMING
PRP	PLATTE RIVER POWER AUTHORITY
PSC	PUBLIC SERVICE COMPANY OF COLORADO
PSI	PUBLIC SERVICE COMPANY OF INDIANA
PSO	PUBLIC SERVICE COMPANY OF OKLAHOMA
PSP	PUGET SOUND POWER AND LIGHT COMPANY
R1	WESTERN AREA POWER ADMIN. REGION 1
R2	WESTERN AREA POWER ADMIN. SACRAMENTO AREA
R3	WESTERN AREA POWER ADMIN. REGION 3
R4	WESTERN AREA POWER ADMIN. SALT LAKE CITY AREA
R5	WESTERN AREA POWER ADMIN. REGION 5
R6	WESTERN AREA POWER ADMIN. BILLINGS AREA
R7	WESTERN AREA POWER ADMIN. DENVER AREA
RCP	RURAL COOPERATIVE POWER ASSOCIATION, (MINNESOTA)
RFT	RAFT RIVER RURAL ELECTRIC COOP
SC	SOUTHERN CALIFORNIA EDISON COMPANY
SCE	SOUTHERN CALIFORNIA EDISON COMPANY
SCL	SEATTLE CITY LIGHT COMPANY
SCP	SOUTHEAST COLORADO POWER ASSOCIATION
SDG	SAN DIEGO GAS AND ELECTRIC COMPANY
SJL	SAINT JOSEPH LIGHT AND POWER COMPANY
SMD	SACRAMENTO MUNICIPAL UTILITIES DISTRICT
SPA	SOUTHWESTERN POWER ADMIN.
SPC	SASKATCHAWAN POWER COMPANY
SPD	SNOHOMISH COUNTY PUBLIC UTILITIES DISTRICT
SPP	SIERRA PACIFIC POWER COMPANY
SPS	SOUTHWESTERN PUBLIC SERVICE COMPANY
SRP	SALT RIVER POWER DISTRICT

continues on next page



Table 2.6.15 – continued from previous page

Ownership Code	Ownership Name
SUB	SPRINGFIELD UTILITY BOARD
SWP	SOUTHWESTERN POWER ADMINISTRATION
SWR	SOUTHWEST REGION - USBR
TCE	TRI-COUNTY ELECTRIC ASSOCIATION, INC. (WYOMING)
TCL	TACOMA CITY LIGHT COMPANY
TEP	TUSCON ELECTRIC POWER COMPANY
TES	TEXAS ELECTRIC SERVICE COMPANY
TGE	TUCSON GAS AND ELECTRIC COMPANY
TIL	TILLAMOOK PUD
TPL	TEXAS POWER AND LIGHT COMPANY
TRI	TRI-STATE GENERATION AND TRANSMISSION ASSOC.
TSG	TRI-STATE GENERATION AND TRANSMISSION ASSOC.
TVA	TENNESEE VALLEY AUTHORITY
UEC	UNION ELECTRIC COMPANY (IOWA,MISSOURI,ILLINOIS)
UPA	UNITED POWER ASSOCIATION, INC. (NORTH DAKOTA)
UPL	UTAH POWER AND LIGHT COMPANY
USN	U.S.NAVY WAP WESTERN AREA POWER ADMINISTRATION-BILLINGS AREA
WEP	WISCONSIN ELECTRIC POWER COMPANY
WIS	WISCONSIN PUBLIC SERVICE CORP.
WKP	WEST KOOTENAY POWER AND LIGHT COMPANY, LTD.
WMP	WISCONSIN MICHIGAN POWER COMPANY
WPD	WHATCOM COUNTY PUD
WPS	WASHINGTON PUBLIC POWER SUPPLY SYSTEM
WRP	WISCONSIN RIVER POWER COMPANY
WRE	WELLS RURAL ELECTRIC CO.
WST	WESTERN POWER AND GAS COMPANY (COLORADO)
WWP	WASHINGTON WATER POWER COMPANY
YWE	YUMA WRAY ELEC.ASSN.,INC.

### 2.6.36 OLD\_BASE

```
/ OLD_BASE, FILE = <filespec>, REBUILD = OFF
                                ON
```

This command specifies that a previously solved Powerflow case is to be loaded from the specified file and used as the base system for the current request.

<filespec> The file specification of the solved network to be re-solved.

The REBUILD switch causes the program to rebuild all of the tables and starts the solution with a “flat start.”

## 2.6.37 OUTAGE\_SIMULATION

/ OUTAGE\_SIMULATION

This command simulates the effect of line outages, load dropping, generator outages, and generator rescheduling. It invokes a process which modifies the base case data in residence. For this reason, this process should not be used with any other process.:

```
/ OUTAGE_SIM
> OLDBASE = filespec
.....
.....
..... Optional Outage Simulation Required Qualifiers
.....
.....
```

where:

filespec File specification for the base file to be loaded to begin the process.

### OUTAGE\_SIMULATION Qualifiers

```
>ANALYSIS = OFF, MINLOADING = 100
ON <num>
```

This command specifies the threshold loading of a line to be included as contingency-caused overloads. This threshold may be raised on individual branches to screen out base case overloads.:

```
>COMMON_MODE, FILE = *
>COMMON_MODE_ONLY, FILE =*
```

These two commands introduce a script which defines one or more “common-mode” outages. The second form restricts the outage simulation study to include only common mode outages. In this case, it is still necessary to introduce an associated >OUTAGE record, but it is used only to define the zones and bases of interest.

The simulation and analysis of any common-mode outages complements in a seamless fashion that for the ordinary single-contingency branch outages. The script associated with the common mode study can be either in a separate file (in which case a file name would be specified) or in the input stream (in which case a file name \* is specified).

Each common-mode outage consists of two parts: A common-mode identification record (>MODE) and the set of WSCC-formatted bus an/or branch changes which are associated with that common mode outage. It is useful and recommended to annotate this script with comment text. An example will illustrate all the points mentioned.:

```
> MODE B/D DRISCOLT 230
B D DRISCOLT 230
. Above common mode outage takes out the following lines:
. DRISCOLT 230 ALLSTON 230
. DRISCOLT 230 CLATSOP 230
. DRISCOLT 230 DRISCOLL 230
.
```

Here, the name of the introduced common-mode outage is “B/D DRISCOLT 230”. The name is arbitrary; it should be sufficiently distinct to contrast it with entities associated with ordinary single contingency branch outages. The name is 40-characters long, is defined in columns (7:47) on the > MODE records, and is truncated to 31 columns in certain Outage Analysis listings where it must compete with the single contingency branch outages. A maximum of 50 >MODE records are permitted.

Each >MODE record is accompanied with an set of arbitrary change records which specifically define all of the changes in the system that are effected by the common mode. If no change records are submitted, the >MODE record is meaningless, since it would not perturb the system in any manner.

The common mode changes permitted are restricted to the following:

- Bus deletion (D) and modification (M).
- Continuation bus deletion (D) and modification (M).
- Branch deletion (D).

The change methodology is identical with that used elsewhere. A bus deletion, for example, automatically deletes all components associated with it. Implementation, however, expedites the change, but in a manner designed to maximize the computational efficiency. The same bus deletion, to repeat the example, is effected by temporarily changing the impedance of all emanating branches to 10000.0 p.u. X, and preserving a small residual admittance for the bus.

With judicious selection of change records, it is possible to simulate complex scenarios such as the loss of switching CAPs followed with the loss of a transformer. If the common mode changes isolate even a single bus in the system, the outage is skipped and the isolation is noted within the outage analysis reports.

Although every attempt was made to simulate common mode outages as efficiently as possible, the highly efficient line compensation schemes that were utilized in the single contingency branch outages could not be used here. Each common mode outage requires refactorization of the associated network matrices.

```
>DEBUG = OFF
      ON
```

This turns on debug dumps.

**Note:** Caution! These dumps can be enormous.

```
>DEFAULT_RATING
```

This command indicates that the following text is line default data. Branches in the specified areas of interest are examined for zero rating (which is an omission of data). If the base kVs of the terminals match the kV in the following default ratings, new ratings are assigned to their branches. They then become candidates for branch overload checking following an outage.

See the table below for the format of default ratings.

This command specifies that all line resistance in the eliminated system is replaced with equivalent current injection. The equivalent network, as modified by this option, is easier to solve.:

```
>EXPAND_NET = 2
      <nn>
```

This command specifies that the border of the selected equivalent network should emanate outwards an additional number of buses. The expansion selected should be less than 100.:

```
>GEN_OUTAGE = NONE
      <nn>
```

This command specifies the maximum number of generator outages for rescheduling.

```
>INCLUDE_CON = <filespec>
```

Use this command to divert the input stream to an auxiliary file that contains / OUTAGE\_SIMULATION text. This auxiliary file cannot contain a recursive / INCLUDE\_CON statement.:

```
>LOW_VOLT_SCREEN = 80
                  <num>
```

This command specifies a contraction for overload values to compensate for effects of voltage changes. It lowers the threshold which tests for overloaded lines.:

```
>MIN_EQUIV_Y = .02
              <num>
```

This command specifies minimum admittance of equivalent branches.:

```
>NO_SOLUTION, ANGLE = 3.0, DELTA_V = .5
                  <num>          <num>
```

This command specifies conditions for no solution (or no convergence). ANGLE is the largest excursion angle (from one iteration to another) in radians relative to the slack bus.

```
>OLD_BASE = <filespec>
```

<filespec> is file specification of the solved network to be re-solved.

Because / OUTAGE\_SIMULATION is a stand-alone process, it must begin by loading an old base file which is introduced using this command.:

```
>OUTAGE, ZONES = *      BASES= *
                <list>  <list>
```

This command specifies the ZONES and voltage levels where outages should be taken. Elements of the list should be separated by commas.:

```
>OUTPUT_SORT , OVER_OUT
                OUT_OVER, OWNER
                BOTH
```

This command specifies the sort order for output listing in terms of overloads and associated outages. The OWNER option requests ownership-bus sort order.:

```
>OVERLOAD, ZONES = *      BASES= *
                <list>    <list>
```

To specify zones and voltage levels where overload should be monitored, use this command. Entities in list should be separated by commas.:

```
>PHASE_SHIFT = FIXED_POWER
                FIXED_ANGLE
```

This command specifies phase-shifter representation constant branch power or constant phase shift angle.:

```
>REACTIVE_SOL = ON
                OFF
```

This command invokes the reactive solution feature. Normally, only the P-constraints are held.:

```
>REALLOCATE = NONE
                LOAD
                LOADGEN
```

This command specifies that load may be shed, generation changed, or both, in order to relieve overload.:

```
>REDUCTION = NONE,REI=OFF
            SIMPLE  ON
            OPTIMAL
```

This command requests the reduction feature and specifies the type of reduction.:

```
>REDUCTION_DEBUG = NONE
                  MINOR
                  MAJOR
```

This is used to request the debug feature.:

```
>RELAX_BR_RATE = ON,PERCENT=5.0
                OFF      <NUM>
```

This command requests that the branch ratings be relaxed by a certain percentage.:

```
>SET_RATINGS, NOMINAL, FILE = <filespec>
                        SUMMER
                        WINTER
```

This command specifies the special ratings of branches that are used for overload determinations. The number of branches whose ratings are specified in FILE is given by a records parameter. Specification records follow this qualifier if the file parameter is omitted. Rating records are described in the table below.

Table 2.6.16: Fields for Rating Records

Column Position	Content
7-15	bus name
16-19	base voltage
21-27	bus name
28-31	base voltage
32	parallel identification
34-37	nominal amps rating
39-42	summer amps rating
44-47	winter amps rating

```
>SOL_ITER, FIXED = 3
                  <nn>
```

This command sets the solution iteration limit per outage. Divergence is assumed when this limit is exceeded.

```
>TOLERANCE = .005
              <num>
```

This specifies the convergence tolerance in per unit power. Convergence is assumed when mismatch is less than this value. Larger values (0.05) yield fast, approximate solutions; smaller values yield slower, more exact solutions.

### Debugging techniques

The following method has proved to be a useful tool for debugging the Outage Simulation Program (OSP) interactively. In invoking this option, three events occur.

1. After the equivalent reduced system is established but before the individual branch outages are taken, the user interactively selects from the full set of branch outages one or more outages. The unselected outages will be ignored.
2. Debugging switches are turned on.
3. Salient process and status information about each outage is displayed on the screen.

This is most useful to confine the study to a single questionable outage which then will be compared with the results of an IPF change case depicting the same outage. To invoke this, enter the two following DCL commands in a terminal window.:

```
$ DEBUG_OUTAGE_SIMULATION_STUDY :== ON
$ RUN IPF_EXE:FSTOUT.EXE_V321
```

The second command executes the OSP interactively. After responding to the prompted Power Flow Control file and waiting a few minutes, OSP's special debugging is invoked.

Enter outage range (n:m), 0=Save, -1=Cancel)

You must select the outage by trial and error using a binary search. Enter a candidate outage branch index (say 127). The dialog continues (using an actual case for an example). 127 outage BELNGMP 115.0 CARILINA 115.0 1 : Select? (Y or N) Selecting "Y" will add this to the outage set "N" will ignore it. If the displayed outage is alphabetically lower than the desired outage, respond with "N" and enter a higher outage number at the next prompt. If it is higher do the opposite. The dialog loops for additional selections or searches.

Enter outage range (n:m), 0=Save, -1=Cancel)

Eventually, when the desired outages(s) is (are) selected, the process is exited with either option ("0", saving the selection and continuing or "-1", ignoring the selection and continuing).

### 2.6.38 OVERLOAD\_RPT

```
/ OVERLOAD_RPT, TX =      90.  , LINE = 90.
                        <num>      <num>
```

This command sets the percentage of line and transformer ratings above which line and transformer loadings are listed in the analysis report.

### 2.6.39 P\_ANALYSIS\_RPT

```
/ P_ANALYSIS_RPT , LEVEL =      2 , *
                                1  ZONES=<list>
                                3  OWNERS=<list>
                                4
```

Use this command to specify the printed analysis report.

When <list> is blank, asterisk or null, ALL is assumed unless limited by a preceding statement.

The level number determines the analysis summaries to be displayed.

For LEVEL=1, the following summaries are included:

- User-defined analysis (optional).
- Buses with unscheduled reactive.

For LEVEL=2, the following are displayed with summaries for LEVEL=1:

- Total system generations and loads by owner.
- System generations, loads, losses and shunts by zones.
- Undervoltage-overvoltage buses.
- Transmission lines loaded above XX.X% of ratings.
- Transformers loaded above XX.X% of ratings.
- Transformer excited above 5% over tap.
- Transmission system losses.
- BPA industrial loads.
- dc system.
- Shunt reactive summary.
- Summary of LTC transformers.
- Summary of phase-shifters.
- Summary of %Var-controlled buses.
- Summary of type BX buses.
- Summary of adjustable Var compensation.
- Transmission lines containing series compensation.

For LEVEL=3, the following is displayed in addition to the LEVEL=2 output:

- Bus quantities.

For LEVEL=4, the following are displayed in addition to the LEVEL=3 display:

- Spinning reserves.
- Transmission line efficiency analysis. Lines loaded above XX.X% of nominal ratings.
- Transformer efficiency analysis. Total losses above X.XX% of nominal ratings.
- Transformer efficiency analysis. Core losses above X.XX% of nominal ratings.

### Example

```
/ F_ANALYSIS_RPT, LEVEL=4, OWNERS= BPA,PGE,PPL,WPS
/ P_ANALYSIS_RPT, LEVEL=1, ZONES = NA, NB, NC
/ F_ANALYSIS_RPT, LEVEL=4, *
/ P_ANALYSIS_RPT, LEVEL=1, ZONES = *
```

## 2.6.40 P\_INPUT\_LIST

```

/ P_INPUT_LIST , NONE
    FULL,  ERRORS = NO_LIST
                LIST
    ZONES = <list>
    ZONES = ALL, FULL, or NONE

```

This command lists input data on PAPER. Output can be restricted to individual zones specified in <list>, which are separated with commas. Note that FULL or NONE may be specified in two forms.

The ERRORS options can be set to NO\_LIST to suppress the input listing if any Fatal (F) errors are encountered.

Example:

```

PWRFL0 case: 9BUS proj: TEST-CASE * * * INPUT LISTING * * * 9 BUSSES 8 EQUIVALENT BR
PAGE 3 14-JUN-94

BASIC NINE-BUS CASE
FOR EXAMPLE REPORTS

BS GEN1 16.5 2 .0PL .0QL .0PS .0QS 240.0PM 306.2PG 150.0QH -100.0QL 1.040VH .0VL
T 1 GEN1 HI 230.0 0 0 MVA 0 C .00000 R .05760 X .00000 G .00000 B 16.50 230.00 0 T 0 E
B 0 B 0IN
B GEN1 HI 230.0 2 .0PL .0QL .0PS .0QS .0PM .0PG .0QH .0QL .000VH .000VL
T 2 GEN1 16.5 0 0 MVA 0 C .00000 R .05760 X .00000 G .00000 B 230.00 16.50 0 T 0 E 0 B
B 0IN
L 2 STA A 230.0 0 0 AMP 0 C .01000 R .08500 X .00000 G .08800 B .0 MI 0 T 0 B 0IN
L 1 STA B 230.0 0 0 AMP 0 C .01700 R .09200 X .00000 G .07900 B .0 MI 0 T 0 B 0IN
BQ GEN2 18.0 1 .0PL .0QL .0PS .0QS 180.0PM 163.0PG 120.0QH -80.0QL 1.025VH .000VL
T 1 GEN2 HI 230.0 0 0 MVA 0 C .00000 R .06250 X .00000 G .00000 B 18.00 230.00 0 T 0 E
B 0 B 0IN
B GEN2 HI 230.0 1 230.0PL .0QL .0PS .0QS .0PM .0PG .0QH .0QL .000VH .000VL
T 2 GEN2 18.0 0 0 MVA 0 C .00000 R .06250 X .00000 G .00000 B 230.00 18.00 0 T 0 E 0 B
B 0IN
L 1 STA A 230.0 0 0 AMP 0 C .03200 R .16100 X .00000 G .15300 B .0 MI 0 T 0 B 0IN
BQ GEN3 13.8 2 .0PL .0QL .0PS .0QS 130.0PM 85.0PG 80.0QH -60.0QL 1.025VH .000VL
T 1 GEN3 HI 230.0 0 0 MVA 0 C .00000 R .05860 X .00000 G .00000 B 13.80 230.00 0 T 0 E
B 0 B 0IN
B GEN3 HI 230.0 2 .0PL .0QL .0PS .0QS .0PM .0PG .0QH .0QL .000VH .000VL
T 2 GEN3 13.8 0 0 MVA 0 C .00000 R .05860 X .00000 G .00000 B 230.00 13.80 0 T 0 E 0 B
B 0IN
L 1 STA B 230.0 0 0 AMP 0 C .03900 R .17000 X .00000 G .17900 B .0 MI 0 T 0 B 0IN
L 1 STA C 230.0 0 0 AMP 0 C .01190 R .10080 X .00000 G .10450 B .0 MI 0 T 0 B 0IN
B STA A 230.0 1 125.0PL 50.0QL .0PS .0QS .0PM .0PG .0QH .0QL .000VH .000VL
L 1 GEN1 HI 230.0 0 0 AMP 0 C .01000 R .08500 X .00000 G .08800 B .0 MI 0 T 0 B 0IN
L 2 GEN2 HI 230.0 0 0 AMP 0 C .03200 R .16100 X .00000 G .15300 B .0 MI 0 T 0 B 0IN
B STA B 230.0 2 90.0PL 30.0QL .0PS .0QS .0PM .0PG .0QH .0QL .000VH .000VL
L 2 GEN1 HI 230.0 0 0 AMP 0 C .01700 R .09200 X .00000 G .07900 B .0 MI 0 T 0 B 0IN
L 2 GEN3 HI 230.0 0 0 AMP 0 C .03900 R .17000 X .00000 G .17900 B .0 MI 0 T 0 B 0IN
B STA C 230.0 2 100.0PL 35.0QL .0PS .0QS .0PM .0PG .0QH .0QL .000VH .000VL
L 2 GEN3 HI 230.0 0 0 AMP 0 C .01190 R .10080 X .00000 G .10450 B .0 MI 0 T 0 B 0IN

PWRFL0 case: 9BUS proj: TEST-CASE * * * INPUT LISTING * * * 9 BUSSES 8 EQUIVALENT BR

```

(continues on next page)



(continued from previous page)

↪ PAGE 3 14-JUN-94

## 2.6.41 P\_OUTPUT\_LIST

```

/ P_OUTPUT_LIST, NONE, FAILED_SOL = FULL_LIST
      FULL          PARTIAL_LIST
      ZONES = <list>    NO_LIST
      ZONES = ALL, FULL, or NONE

```

This command lists output on PAPER. Output can be restricted to individual zones specified in <list>, which are separated with commas. Note that FULL or NONE may be specified in two forms.

The FAILED\_SOL option is set to override the output listing if a failed solution occurs. It defaults to a full listing. A PARTIAL\_LIST observes zone lists.

Example:

```

PWRFL0 case: 9BUS proj: TEST-CASE * * * * DETAILED OUTPUT LISTING * * * * PAGE 8 14-JUN-
↪ 94

BASIC NINE-BUS CASE
FOR EXAMPLE REPORTS

A COMPLETE OUTPUT LISTING OF ALL BUSSES WILL BE GIVEN

GEN1    16.5   17.2KV/ .0 ZONE 2 306.2PGE  72.5QCOND  .0PLOAD  .0QLOAD    1.040PU_
↪ KV      BUS TYPE S      GEN1 HI   230.0                2 306.2PIN  72.5QIN      _
↪ 0.0PLOSS  52.7QLOSS          16.5/230.0
↪ 306.2PNET 72.5QNET                                0.0 SLACK ADJ

```

## 2.6.42 REBUILD

```

/ REBUILD = OFF
      ON

```

This command requests that all internal data tables be rebuilt using the current specified OLDBASE file. This has the same effect in a case as the REBUILD parameter on the / OLD\_BASE statement.

## 2.6.43 REDUCTION

```

/ REDUCTION

```

This command reduces the network in residence to a desired size and solves the reduced network. It can be saved or processed further as an ordinary base case. For more detail on the methods used, see [Network Reduction.](#):

```

/ REDUCTION
.....
.....
..... Optional Reduction Qualifiers

```

(continues on next page)

(continued from previous page)

```
.....
.....
```

## Reduction Qualifiers

>COHERENT\_CLUSTERS, <name> <base kV>

This identifies row-coherent generators (or load) of an REI subsystem. The name must be unique, containing 1-7 characters without blanks and be left-justified. The REI components, which will have their generation and/or load transferred to the coherent generator, are identified with ordinary WSCC-formatted bus (Type B) records which follow.

The named constituent buses which comprise each coherent cluster may be either retained or eliminated buses. In either case, the constituent buses will be eliminated.

Special codes on each bus permit individual dispositions of generator and load quantities. Generation and/or load may be converted to constant current, constant admittance, or converted to an REI coherent unit. The codes are shown in the table below.

Table 2.6.17: Reduction Qualifier Codes

Column	Value
3 (Generation)	0 - Constant Current
3 (Generation)	1 - Constant Admittance
3 (Generation)	Blank or 2 - REI
4 (Load)	0 - Constant Current
4 (Load)	1 - Constant Admittance
4 (Load)	Blank or 2 - REI

```
>DEBUG = NONE
        MINOR
        MAJOR
        ORDERING
```

Use this to request the debug feature.

Table 2.6.18: Reduction Debug Codes

Debug	Effect
Minor	Minimal debug.
Major	Includes dump of intermittent reduction steps.
Ordering	Includes full dump of reordering arrays during each nodal elimination step.

```
>ELIM_MODE, GEN = CURRENT, LOAD = CURRENT, SHUNT_Y = ADMITT
                ADMITT      ADMITT      CURRENT
                REI,PMIN=<n>  REI          REI
```

This command determines how the nodal generation, load, and shunt admittance on eliminated nodes is to be processed. It does not affect the original quantities of the interior or envelop (border) nodes. The disposal options are to convert selected quantities to nodal current, to nodal shunt admittance, or to append them to an REI node.

>ENVELOPE\_BUSES = BE

This command, when elected, changes the subtypes of all envelope node to type BE. Its primary merit is to secure the voltages of the terminal buses at their base case values and improve the solvability of the reduced equivalent system. The default option is to leave the envelope buses in their original subtype.

```
>EXCLUDE_BUSES
```

This command excludes from the retained network the buses listed on the bus-formatted records following this statement. Its purpose is to allow more flexibility in the definition than allowed with a simple SAVE\_BASES or SAVE\_ZONES. Obviously, the retained system must already be defined by a prior SAVE\_BASES or SAVE\_ZONES command.

```
>INCLUDE_BUSES
```

This command includes in the retained network additional buses listed on the bus-formatted records following this statement. Its purpose is to allow more flexibility in the definition than allowed with a simple SAVE\_BASES or SAVE\_ZONES. Obviously, the retained system must already be defined by a prior SAVE\_BASES or SAVE\_ZONES command.

```
>INCLUDE_CON = <filespec>
```

Use this command to include a set of user-specified default command qualifiers, which is stored in a file. Such a default command file should not contain this / INCLUDE\_CON statement.:

```
>KEEP_AI_SYS = ON
               OFF
```

This command requests that the equivalent network will retain all of the attributes of area interchange control. This includes all area slack nodes and all tie line terminal nodes.:

```
>MIN_EQUIV_Y = .02
               <num>
```

This command specifies the minimum admittance of equivalent branches that are retained. Its purpose is to reduce the large number of equivalent branches which are generated, some of which have such large impedances that their contribution to the flows are marginal. A smaller value of 1.0 is recommended. Equivalent branches which have lower admittances (or what is the same, higher impedances) will be replaced with equivalent shunt admittances at both terminals.:

```
>OPTIMAL_REDU = ON
               OFF
```

This command switches the optimal network determination feature, which precedes the actual network reduction. When the optimal network selection is ON, it may enlarge the user-specified retained system with optimally selected nodes such that the overall size of the reduced system will be minimized. In essence, it expands the boundary into the eliminated system in a manner which will topologically result in an equivalent network having more buses but fewer branches overall. Thus, the user defines a fuzzy retained system containing the minimum desired configuration, and the optimal network selection will enlarge the network if feasible.:

```
>RETAIN_GEN = OFF, PMIN = 100.0
               ON         <num>
```

This command selected all generators with generation > PMIN to be in the retained network.:

```
>REI_CLUSTERS, VOLT_DIFF = .25, ANGLE_DIFF = 20.
                  <num>         <num>
```

This command works in conjunction with the REI option on the ELIM\_MODE command. An attempt is made to automatically consolidate REI clusters which may have only a single node. However, their consolidation may result in an equivalent REI node whose voltages are too bizarre. It is electrically correct, but may cause solution problems since voltages are initialized about 1.0. By restricting the voltage differences of REI consolidation candidates to those

whose voltage differences are less than the user-prescribed value, the resultant consolidated REI cluster will have a more feasible voltage.:

```
>SAVE_BASES = <list>
```

This command defined the retained network as consisting of those buses which have the base kvs in list. Elements of list are separated with commas (,).:

```
>SAVE_BUSES
```

This command defines the retained network as consisting of all buses identified on the following bus-formatted records. It is a brute force method to define the retained network. It cannot be used in conjunction with SAVE\_ZONES or SAVE\_BASES. See INCLUDE\_BUSES.:

```
>SAVE_ZONES = <list>, BASES = <list>
```

This command defines the retained network as consisting of those buses which have zones in the first list, with the optional, additional provision that their base kvs must be in the second list. Elements of the list are separated with commas (,).:

```
>STARTING_VOLTAGES = FLAT
                     HOT
```

This command defines the starting voltages which will be used in the ensuing rebuilding and solution of the reduced equivalent base. The default is FLAT, meaning that the solution will use flat starting voltages. There are two separate applications for this option. The first application is to verify the integrity of the equivalent bus and branch data structures from the complex reduction processing. When used in conjunction a another solution option:

```
/ SOLUTION > BASE_SOLUTION
```

the ensuing convergence checks performed in output report independently verify the validity of the reduced bus and branch data. The second application is to assist in a solution of a reduced equivalent system if such assistance becomes necessary.:

```
>ULT_MODE, GEN = CURRENT, LOAD = CURRENT, SHUNT_Y = ADMITT
                ADMITT      ADMITT      CURRENT
                POWER        POWER        POWER
```

This command defines the ultimate form which the currents distributed from the eliminated nodes to the border nodes will attain. It affects only the border nodes. Note that before the elimination, the generation, load, and shunt of each eliminated node is disposed as defined by the command ELIM\_MODE. Those quantities, which were distributed as three separate current vectors during the network reduction, are now to be transformed into their ultimate form. The distributed currents (generation, load, and shunt) will be encoded into special types of +A continuation buses with ownership \*\*\*.

Table 2.6.19: Ultimate Form of Distributed Currents

Option	Meaning	Code year	Comment
CUR-RENT	Constant current model	01	The load elds are interpreted as constant current, constant power factor
ADMITT	Constant admittance	01	The shunt elds are interpreted in the ordinary manner.
POWER	Constant MVA	02	The generation elds are interpreted in the ordinary manner.

It should be noted that the special continuation records +A with ownership \*\*\* will always be generated to hold the equivalent shunt admittance which results from the admittance to ground in the eliminated system.

## 2.6.44 RPT\_SORT

```
/ RPT_SORT = BUS
           ZONE
           AREA
           OWNER
```

This command sorts output information of a solved network by bus, zone, area, or ownership. The area sort is by A0 records, not by A records. See section *Area Output Sort (AO)*.

## 2.6.45 SAVE\_FILE

```
/ SAVE_FILE, TYPE = WSCC_ASCII, FILE = <filespec>
           WSCC_BINARY

/ SAVE_FILE, TYPE = NEW_BASE, FILE = <filespec>

/ SAVE_FILE, TYPE = NETWORK_DATA, FILE = <filespec>,
           DIALECTS = BPA,
                       WSCC,
                       WSCC1,
                       PTI,
           SIZE = 120,
                80,
           RATINGS = EXTENDED
                   MINIMUM
                   NOMINAL

/ SAVE_FILE, TYPE = CHANGES, FILE = <filespec>
```

These commands request that the identified file type be written to the named file. Type = WSCC\_ASCII or type = WSCC\_BINARY writes an interface file which can be read by the WSCC Stability Program (version 9 or greater) in lieu of an IPS history file. The filename must be specified. The file can be written in either formatted ASCII or unformatted binary format. The binary format is more compact, but the ASCII file can be freely transferred between platforms with unlike hardware and/or operating systems. The file contains only that powerflow information which is required by the Stability Program; it is not a complete base case. Type = NEW\_BASE is identical in function to the command /NEW\_BASE, file = <filename> Type = NETWORK\_DATA writes the complete network data file in various WSCC-formatted dialects.

- The BPA dialect writes the network data in the form most identical to its originally submitted form.
- The WSCC dialect ignores Interarea “I” records, consolidates all “+” bus records (with the exception of +A INT records) with the associated B-record; writes types L,E,T,TP,LM, and RZ branch records in the order of their original submittal; writes type R records in the order adjustable tap side to fixed tap side, or hi-low; writes type LD records in the order rectifier-inverter; writes all branch data with a minimum of  $X = 0.0005$  p.u.; sets Vmin on bus types BV, BX, BD, and BM to 0.0, sets non-zero Qmin on bus types B , BC, BV, and BT to zero; changes type BE buses with non-zero Q-limits to type BQ; and changes zero Qmin and Qmax on type BE buses to type B.
- The WSCC1 dialect includes all of the WSCC dialect mentioned above, and includes consolidating all branches consisting of sections into a single equivalent branch.
- The PTI dialect ignores Interarea “I” records, consolidates all “+” bus records with the associated B-record; sets Vmin on bus types BV, BX, BD, and BM to 0.0, sets non-zero Qmin on bus types B , BC, BV, and BT to zero;

changes type BE buses with non-zero Q-limits to type BQ; and changes zero Qmin and Qmax on type BE buses to type B. writes types L, E, T, TP, and RZ (ignores type LM) branch records in the order of their original submittal; writes type R records in the order adjustable tap side to fixed tap side, or hi-low; writes type LD records in the order rectifier-inverter.

In addition, type = NETWORK\_DATA writes the MINIMUM or NOMINAL branch current rating in the NOMINAL field if that option is selected; the EXTENDED ratings are written to columns 81:92 only if the BPA dialect, the EXTENDED rating, and the 120-character record size are all selected (all are defaults). Type = CHANGES writes the complete set of network changes to the named file.

## 2.6.46 SOLUTION

/ SOLUTION

This command enables solution options and special post-solution processes.:

```
>AI_CONTROL = CON
              MON
              OFF
```

This command sets the switches for area interchange to CONtrol, MONitor, or OFF.

The alternate voltages and LTC taps are encoded on type B and T records. See Figure 4-13 and Figure 4-14.

### SOLUTION Qualiers

>BASE\_SOLUTION

This command completely bypasses the solution routine and uses the base voltages in residence. It is useful for debugging purposes, such as validating Network Reduction, or for examining the actual old solution quantities directly from a base case.:

```
>DEBUG, TX=OFF, BUS=OFF, AI=OFF, DCMODEL=OFF
          ON      ON      ON      ON
```

This command turns on the following various program debug switches. See the table below.

Table 2.6.20: Debug Switches

Switch	Meaning
TX:	LTC
BUS:	Bus Switching
AI:	Area Interchange
DCMODEL:	DC Modeling

```
>LIMITS, QRES= 0.10 ,PHA= 45.001,DEL_ANG= 1.000, DEL_VOLT= .150
              <num>      <num>      <num>      <num>
```

To set limits, the statement >LIMITS may be used as many times as is needed.

QRES p.u. MVAR by which a BQ, BG, or BX bus must be perturbed to revert from a state of Q-max control to a state of V control.

PHA Minimum angle in degrees for which fixed-tap phase shifters are modeled as ideal (no loss) devices in the decoupled starting routine.

DEL\_ANG Maximum angle adjustment in radians permitted in one Newton-Raphson iteration.

DEL\_VOLT Maximum voltage adjustment in per unit permitted in one Newton-Raphson iteration.

```
> LOAD_SOLUTION, VOLTAGES =RECTANGULAR, FILE = file_name
                        POLAR
                        DEBUG = OFF, SOLUTION = BASE
                        ON      HOTSTART
```

>LOAD\_SOLUTION loads an alternate set of voltages and LTC taps for either the base solution (SOLUTION = BASE) or for a hot start (SOLUTION = HOTSTART). The purpose of this command is to validate other Powerflow programs (PTI, SVSPP) using similar base case data or to assist difficult solutions by providing an alternate starting point. The contents of data in file\_name are shown below.

B										NAME	kV	VMAG										ANGLE																		
												e										f																		
	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
B																																								
B																																								
B																																								

Fig. 2.6.12: Alternate Voltages and LTC Taps for B Records

T										NAME1									kV 1																		NAME2									kV 2									C	S	E	C	T	I	O	N										TAP																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														
										1																		2																		3																		4																		5																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5

Fig. 2.6.13: Alternate Voltages and LTC Taps for T Records

The formats of the voltages and taps are “free-field,” meaning that the information must begin at least in the column noted and that additional entities are separated with a blank space.

On the T record, tap is in per unit (TAP1/BASE1)/(TAP2/BASE2) or in radians for an LTC phase shifter.

### Restrictions

The following restrictions apply:

- All sets of B records must include all buses.
- The voltages must be within global limits.
- Only LTC transformers can have tap changes.
- The taps must be within LTC tap limits.

```
>LTC = ON
      ON_NV
      ON_NPS
      OFF
      ON_DCONLY
```

This is used with the following to set the control of LTC transformers.

ON Full LTC control.

ON\_NV Partial LTC control (P and Q only).

ON\_NPS Full LTC voltage control, no LTC phase shifter control.

OFF No LTC control.

ON\_DCONLY No LTC control (except for DC commutating transformer).

---

**Note:** To obtain meaningful results with the options ON\_NV, ON\_NPS, or ON\_DCONLY, the taps of the other LTC's should not change. Recall that normally with a flat start (VFLATSTART=ON) all LTC's, including those turned off, start at the mid tap value. To represent this action, invoke the additional command:

```
>MISC_CNTRL, VFLATSTART =ON, DCLP = ON, X_BUS = BPA,
```

```
      OFF OFF WSCC
      ITER_SUM = OFF, TSTART =0.5
      ON
      NUMVSTEPS = 3 PHASE_SHIFTER_BIAS = BPA
      WSCC
      BRIDGE_CURRENT_RATING = ON
      OFF
```

---

These commands and options set the solution controls.:

```
X_BUS = BPA
      VMAX
      WSCC
```

Solution option for type BX buses. The BPA option accepts any discrete reactance step on a BX bus when its solution voltage  $V$  lies in the range of  $V_{min} \leq V \leq V_{max}$ . The VMAX option attempts to find the switched reactance step on each BX such that its solution voltage is the largest voltage  $V \leq V_{max}$ . The WSCC option adjusts discrete steps on a BX bus (using as an initial value the B\_shunt entered on the input record) only when the voltage violates the limits  $V_{min}$  or  $V_{max}$ .



## DCLP

DC solution technique for multi-terminal DC lines. OFF uses old version. ON uses linear programming (LP) routines — default is ON, and it should not be changed except in extraordinary circumstances.

**VFLATSTART = ON**

OFF`

**Starting option. ON = flat start. Initialization under flat start invokes the following conditions.**

- All voltages are set  $V + j0$  where  $V$  is either the controlled voltage or a value between  $V_{min}$  and  $V_{max}$ .
- All special bus types BV, BQ, BG, and BX are in nominal state. (See options X\_BUS.)
- All LTCs are set to a uniform starting value. All LTS phase shifters are set to a value closest to zero, but within top limits. (See option TSTART.)
- All DC quantities invoke AC terminal conditions.

TSTART = 0.50

This sets the LTC transformer starting tap. The starting tap is based on this value together with the relative magnitude of the variable-tap side and fixed-tap side base kVs.

If Base\_variable\_tap\_side < Base\_fixed\_tap\_side then

Tap = (1-TSTART) \* Tmin + TSTART \* Tmax

Otherwise, it is

Tap < TSTART \* Tmin + (1-TSTART) \* Tmax

**ITER\_SUM = OFF**

ON`

This switch controls printout of the solution iteration detail report.

OFF = Print out report only if a failed solution occurs.

ON = Print out report unconditionally.

**PHASE\_SHIFTER\_BIAS = BPA**

WSCC`

This switch determines the phase shifter angle bias for type RM phase shifters. The BPA option biases the angle to zero degrees (in recognition that non-zero degree phase shifts cause real-power loop flow and that biasing the angle to zero effectively bias the system to lower losses since the additional losses in the loop are minimized). The WSCC option biases the angle to the original angle specified on the TP record):

BRIDGE\_CURRENT\_RATING = ON OFF

This switch determines whether the converter bridge current rating is applied to the DC circuit rating. ON implies that the bridge current rating does apply; OFF implies that it doesn't.

```
>SOL_ITER, DECOUPLED = 2, NEWTON = 30
               <n>      <nn>
```

This command sets the solution and iteration limits with the following:

DECOUPLED Uses decoupled method to iterate from flat start. INITIAL prefixes the solution iteration count.

NEWTON Uses Newton-Raphson method. At least three iterations must be specified. This is the final solution.

```
>TOLERANCE, BUS = 0.001, AIPOWER = 0.001, TX = 0.001, Q = 0.001, V = 0.001
               <num>          <num>          <num>          <num>          <num>
```

This command sets the tolerances in per unit (p.u.) for convergence testing. BUS pertains to both  $P_{net}$  and  $Q_{net}$  (in per unit), AI\_POWER to net area interchange export (in per unit), TX to Pkm for type RP or RM LTC phase shifters or for Qkm for type RQ and RN LTC transformers, Q to the violation of  $Q_{max}$  or  $Q_{min}$  for special bus types BQ, BG, and BX, and V to the desired controlled voltage by type BG buses or by type R transformers.

### 2.6.47 SORT\_ANALYSIS

```
/ SORT_ANALYSIS , OVERLOAD = < BUS >, OVERVOLTAGE = < BUS >
                                < OWNER >          < OWNER >
                                < ZONE >           < ZONE >
                                < AREA >          < AREA >
```

This controls the sort order of selected analysis listings:

- Overloaded transmission lines.
- Overloaded transformers.
- Undervoltage/overvoltage buses.

The defaults are determined by RPT\_SORT options:

- Sort by <ZONE> if RPT\_SORT = <ZONE>.
- Sort by <AREA> if RPT\_SORT = <AREA>.
- Sort by <OWNER> if RPT\_SORT is defaulted.

### 2.6.48 TRACE

```
/ TRACE, REORDER = OFF, X_REF = OFF, AUTO = OFF, Y_MATRIX = OFF, -
                   ON          PART      ON          ON
                   FULL
                   OUTPUT = OFF, MERGE = OFF, CHANGE = OFF
                   ON          ON          ON
```

Use this command to monitor data used by the program in various functional applications. This is an aid to user data verification.

### 2.6.49 TRANSFER\_SENSITIVITIES

```
/ TRANSFER_SENSITIVITIES
> OUTAGE
L ...
E ... 1 to 100 branch records in WSCC format identifying outages
T ...
> OVERLOAD
L ...
E ... 1 to 100 branch records in WSCC format identifying lines
```

(continues on next page)

(continued from previous page)

```

T ...
checked for overloads.
> TRANSFER
I ...
I ... 1 to 100 intertie records identifying transfer between
I ... two areas which will alleviate any line overloads.
I

```

This feature is similar to the outage simulation analysis, but with an important difference. Each overloaded line resulting from a contingency is quantified as to the amount of area transfer (shifting of generation from one area to another) that is necessary to alleviate the overload.

Because the problem contains three nested loops (contingency loop, line overload monitoring loop, and inter-area transfer loop), the inter-area transfer loop is restricted to user-specified sets of transfers. Even with the present limits, the number of transfers can be large (100 outages \* 100 overloads \* 100 transfers = 1,000,000).

Limitations and restrictions.

- Each / TRANSFER\_SENSITIVITIES must include all three components:  
     > OVERLOAD > OUTAGE > TRANSFER
- Each > OVERLOAD, > OUTAGE, and > TRANSFER component must be followed with relevant branch records of type L, E, or T, or relevant intertie records of type I.

The individual limits are:

Limit of OVERLOAD branches: 100 Limit of OUTAGE branches: 100 Limit of inter-area TRANSFER: 100

- Lines monitored for overload must have rating (nominal) > 0.0. Extended ratings are not used.
- Lines that are candidates for overloads or outages must have  $R < X$ .

## 2.6.50 TX\_EFF

```

/ TX_EFF, TOTAL_LOSS = <0.04>, CORE_LOSS = <0.02>, OWNERS = < BPA >
                        <nn>                      <nn>                < list >

```

Use this command to compare total and core transformer losses. The output can be filtered by owners. “BPA” is the default if no owners are specified.

## 2.6.51 USER\_ANALYSIS

```

/ USER_ANALYSIS, FILE = <file_name>, DEBUG = OFF, OUTPUT = <filename>
                                ON

```

This command generates customized analysis listings.

USER\_ANALYSIS provides a simple macro-like programming language to perform algebraic operations involving quantities available or used in the base case in residence. This capability is not related in any way to the CFLOW programming library, which is much more powerful and flexible. However, USER\_ANALYSIS provides an easy way to generate simple reports without C programming.

If FILE in the above command is omitted, or <file\_name> is \*, the user-defined text follows in the input stream. Otherwise, the named file becomes the input stream for this command.

DEBUG enables the debug switch. Its output appears in an output file with the extension .pfd.

OUTPUT = <filename> places a copy of the user defined output into the file selected in <filename>.

The following quantities are available from the solved base case in residence:

- Line flows, P\_in or Q\_in, measured at either bus1 or bus2 terminal.
- Intertie flows, P\_in or Q\_in, measured at metering points.
- Scheduled intertie flow, P\_sched.
- Losses by Zones, Ownerships, Areas or system totals.
- All bus quantities: P\_gen, P\_max, Q\_gen, Q\_max, Q\_min, P\_load, Q\_load, Caps\_scheduled, Caps\_used, Reactors\_scheduled, Reactors\_used, Susceptance\_used, Susceptance\_scheduled, Q\_unscheduled. Voltage in per unit or in kV.

### Example

A simple example illustrates this concept. The records following the / USER\_ANALYSIS have not been introduced. However, the scheme is simple, and the purpose of the records is fairly obvious.

In the example, it is desired to list the sums (P\_in and Q\_in) of two selected branches.

```

/ USER_ANALYSIS, FILE = *
.
. The following symbols define P_in.
.
> DEFINE_TYPE BRANCH_P
LET A1 = ELDORADO 500*LUGO          500 1
LET A2 = MOHAVE    500*LUGO          500 1
> DEFINE_TYPE FUNCTION
LET S1 = A1 + A2
.
. The following symbols define Q_in.
.
> DEFINE_TYPE BRANCH_Q
LET B1 = ELDORADO 500*LUGO          500 1
LET B2 = MOHAVE    500*LUGO          500 1
> DEFINE_TYPE FUNCTION
LET S2 = B1 + B2
H
C Branch                P_in    Q_in
C                        (MW)    (MVAR)
C
C ELDORADO/LUGO          500. 1 = $A1    $B1
C MOHAVE /LUGO           500. 1 = $A2    $B2
C
C Total                  = $S1    $S2

```

Notice that this example contains two types of data: definitions and comments.

The definitions > DEFINE\_TYPE BRANCH\_P, > DEFINE\_TYPE BRANCH\_Q, and > DEFINE\_TYPE FUNCTION identify the type of symbols that follow.

The symbols A1, A2, S1, B1, B2, and S2 are assigned to specific quantities in the network. They are evaluated after the case is solved.

In the analysis phase, the user-defined report is compiled first. The report consists of a single pass through the comment text, substituting symbols for evaluated quantities before the line is printed. The symbols are identified with a leading \$ followed by a valid symbol name.

Let us make the assumption that the symbols above are evaluated as follows:

```
A1 = 859.2
A2 = 901.8
S1 = 1761.0
B1 = 245.1
B2 = 254.2
S2 = 499.3
```

Immediately before printing, these values are encoded into the symbol fields into the comment text. Their default format is F6.0. The output report appears as follows.:

Branch		P_in (MW)	Q_in (MVAR)
ELDORADO/LUGO	500. 1 =	859.	245.
MOHAVE /LUGO	500. 1 =	902.	254.
Total	=	1761.	499.

Three types of text follow the / USER\_ANALYSIS record: pagination specifications, symbol definitions, and user-defined comment text. They are described in more detail in the following sections and are then illustrated with a second example.

## Symbol Definitions

Symbol definitions have either a one-line or multiple-line format.

The one-line format is:

```
> DEFINE_TYPE <symbol_type><symbol_name>=<id_of_computed_quantity>
```

The multiple-line format is:

```
> DEFINE_TYPE <symbol_type> <symbol_name> = <id_of_computed_quantity> <symbol_name> =
-><id_of_computed_quantity> <symbol_name> = <id_of_computed_quantity>
```

Some simple rules must be followed:

- All > DEFINE\_TYPE data is free-field. Blanks and commas are delimiters. If a blank character is part of a name, substitute a pound sign (#).

Example: JOHN DAY 500.0 --> JOHN#DAY 500.0

A blank circuit ID in a multicircuit line must also be entered as a pound sign (#).

Example: B = ELDORADO 500 LUGO 500 #

- All > DEFINE\_TYPE data is case-insensitive. No distinction is made between upper and lower case symbol characters. The case, however, is preserved in the analysis report as it was entered.
- The symbol names are limited to six characters.
- The symbol quantities are encoded with a default format of F6.0. The field begins in the column position of the \$ and continues the necessary field width (default is six) as specified by the format. This default format may be changed by appending the new format to the symbol name using the FORTRAN convention, as shown below.

$$C_{Total} = \$S1/F8.1 \quad \$S2/F8.1$$

### >DEFINE\_TYPE BRANCH\_P and >DEFINE\_TYPE BRANCH\_Q

This defines line flows, both  $P_{in}$  and  $Q_{in}$ , computed at the bus1 terminal (default) or at bus2 if an asterisk (\*) immediately follows the base kV. Below, square brackets “[” and “]” denote enclosed optional quantities.:

```
>DEFINE_TYPE BRANCH_P (Branch flow in MW)
LET P1 = BUS1 BASE1[*] BUS2 BASE2[*] ID
LET P2 = BUS3 BASE3[*] BUS3 BASE4[*] ID
...
LET PN = BUSM BASEM[*] BUSN BASEN[*] ID
>DEFINE_TYPE BRANCH_Q (Branch flow in MW)
LET Q1 = BUS1 BASE1[*] BUS2 BASE2[*] ID
LET Q2 = BUS3 BASE3[*] BUS3 BASE4[*] ID
...
LET QN = BUSM BASEM[*] BUSN BASEN[*] ID
```

### >DEFINE\_TYPE INTERTIE\_P or DEFINE\_TYPE INTERTIE\_Q

This defines the area interchange flow, either  $P_{in}$  (MW) or  $Q_{in}$  (MVAR). Either quantity is computed at the tie-line metering points.:

```
>DEFINE_TYPE INTERTIE_P (Interchange flow in MW)
LET I1 = AREA_1 AREA_2
LET I2 = AREA_3 AREA_4
...
LET IN = AREA_M AREA_N
>DEFINE_TYPE INTERTIE_Q (Interchange flow in MVAR)
LET J1 = AREA_1 AREA_2
LET J2 = AREA_3 AREA_4
...
LET JN = AREA_M AREA_N
```

### > DEFINE\_TYPE INTERTIE\_P\_SCHEDULED

This defines the scheduled area intertie flow (I records) as  $P_{in}$  (MW).:

```
>DEFINE_TYPE INTERTIE_P_SCHEDULED (Scheduled Interchange flow in MW)
LET I1 = AREA_1 AREA_2
LET I2 = AREA_3 AREA_4
...
LET IN = AREA_M AREA_N
```

**>DEFINE\_TYPE OWNER\_LOSS AREA\_LOSS ZONE\_LOSS SYSTEM\_LOSS**

```
>DEFINE_TYPE OWNER_LOSS
LET O1 = BPA, O2 = PGE, etc.
>DEFINE_TYPE AREA_LOSS,
LET A1 = NORTHWEST, etc.
>DEFINE_TYPE ZONE_LOSS,
LET Z1 = NA, Z2 = NB, etc.
>DEFINE_TYPE SYSTEM_LOSS,
LET SYSTOT
```

This defines losses by Area, Zone, Ownership, or total system.

**>DEFINE\_TYPE FUNCTION**

This defines the following records as containing algebraic operators:

+, -, \*, /, \*\*, <, >, (, and ).

```
>DEFINE_TYPE FUNCTION
LET T1 = (Z1 ** 2 + Z2 ** 2 ) ** 0.5
LET T2 = (Z1 > 1200.0) * 100.0
```

T1 evaluates as the square root of the sum of the squares Z1 and Z2. T2 evaluates as 100 times the excess of Z1 over 1200.0. Some simple rules must be followed:

- All symbols referenced on the right-hand side must be defined prior to reference.
- Parentheses can be nested to any level. Operation begins inside the innermost level.
- A single function is limited to 30 symbols and operators.
- Operators have the following precedence (highest to lowest):

```
**
*,/
>,<
+,-
```

FUNCTION admits simple trigonometric (and one absolute value) functions where the relevant arguments or returned values are expressed in radians:

```
sin(), cos(), tan(), arcsin(), arccos(), arctan(), and abs().
```

Here is an example: compute the voltage angle difference between two buses, KEELER 500 and PAUL 500.

```
> DEFINE_TYPE BUS_INDEX
LET A = KEELER 500
LET B = PAUL 500
> DEFINE_TYPE FUNCTION
LET DIF = 57.29578 * (ARCTAN(A.VI/A.VR) - ARCTAN(B.VI/B.VR))
```

Here, A.VR and A.VI are the real and imaginary components to the per unit voltage at KEELER 500. Similarly, B.VR and B.VI for PAUL 500. The ARCTAN function returns the angle in radians. The difference is then converted to degrees.

### >DEFINE\_TYPE OLDBASE

This defines pertinent information from the retrieved OLD\_BASE data file. An example will demonstrate the use of these symbols.

```
>DEFINE_TYPE OLDBASE
LET A = DISK
LET B = DIR
LET C = FILE
LET D = CASE
LET E = DATE
LET F = TIME
LET G = DESC
LET H = PFVER
LET I = USER

C OLD_BASE CASE = $D/A10
C DESCRIPTION $G/A20
C
C GENERATED ON $E/A10 $F/A10
C OWNER $I/A10
C POWERFLOW VERSION $H/A10
```

### > DEFINE\_TYPE BUS\_INDEX

This defines the following records as bus indices. This index is used in conjunction with a coded suffix to obtain specific bus quantities.

Valid suffixes and their associated bus quantities are shown the table below.



Table 2.6.21: Suffixes for Bus Quantities

Suffix	Quantity
.PL	P_load in MW
.QL	Q_load in MVAR
.PG	P_gen in MW
.PM	P_max in MW
.QG	Q_gen in MVAR
.QM	Q_max in MVAR
.QN	Q_min in MVAR
.RKK	Real part of driving point admittance ( $Y_{KK}=R_{KK}+jX_{KK}$ ). Also known as short circuit admittance.
.XKK	Imaginary part of driving point admittance ( $Y_{KK}=R_{KK}+jX_{KK}$ ). Also known as short circuit admittance.
.V	V in per unit
.VA	Voltage angle in degrees
.VR	V in per unit, real component
.VI	V in per unit, imaginary component
.VK	V in kV
.VM	V_max in per unit
.VN	V_min in per unit
.C	Q_caps used in MVAR
.CM	Q_caps scheduled in MVAR
.R	Q_reactors used in MVAR
.RM	Q_reactors scheduled in MVAR
.QU	Q_unscheduled in MVAR
.DVQ	dV/dQ sensitivity kV/MVAR
.DVP	dV/dP sensitivity in kV/MW.
.S	Total reactive used (Capacitors or Reactors) in MVAR.
.SM	Total reactive available (Capacitors or Reactors) in MVAR

An example will demonstrate these concepts.

Compute the generator current (in amps) of Paul 500.0.

```
> DEFINE_TYPE BUS_INDEX
LET A = PAUL 500.0
> DEFINE_TYPE FUNCTION
LET B = (A.PG ** 2 + A.QG ** 2) ** 0.5
LET C = 1000.0 * B / (3.0 ** 0.5 * A.VK)
```

Here symbol B contains the generation in MVA, and C contains the current in amps.

### > DEFINE\_TYPE BRANCH\_INDEX

This defines the following records as branch indices. This index is used in conjunction with a coded suffix to obtain specific branch quantities.

Valid suffixes and their associated branch quantities are shown in the table below.

Table 2.6.22: Suffixes for Branch Quantities

Suffix	Quantity
.TAP1	Tap1 in kV for a T or in degrees for a TP record
.TAP2	Tap2 in kV for a T or TP record
.TAP	The discrete tap number (lowest tap = 1) for an LTC transformer.
.TAPS	The total number of discrete taps for a LTC transformer.

An example will demonstrate these concepts.

Show the tap, the discrete tap number, and the number of discrete taps for a transformer FRANKLIN 115/230

```
>DEFINE_TYPE LINE_INDEX
LET FR = FRANKLIN 115.0 FRANKLIN 230.0
C TX AT      TAP      TAP #      # OF TAPS
C FRANKLIN $FR.TAP2/F7.3 $FR.TAP/F3.0 $FR.TAPS/F3.0
```

Here symbol B contains the generation in MVA, and C contains the current in amps.

### > DEFINE\_TYPE ZONE\_INDEX

This defines the following records as zone indices. This index is used in conjunction with a coded suffix to obtain specific zone quantities.

Valid suffixes and their associated zonal quantities are shown in the table below.

Table 2.6.23: Suffixes for Zonal Quantities

Suffix	Zonal Quantity
.PG	P_gen in MW
.QG	Q_gen in MVAR
.PL	P_load in MW
.QL	Q_load in MVAR
.PLS	P_loss in MW
.QLS	Q_loss in MVAR
.PSH	Installed (Scheduled) P_shunt in MW
.QSH	Installed (Scheduled) Q_shunt in MVAR
.SCAP	Installed (Scheduled) Q_cap in MVAR
.SREK	Installed (Scheduled) Q_reactors in MVAR
.UCAP	Used Q_cap in MVAR
.UREK	IUsed Q_reactors in MVAR

The following example illustrates these concepts.

```
> DEFINE_TYPE ZONE_INDEX
LET ZA = NA
LET ZB = NB
LET TP = ZA.PLS + ZB.PLS
LET TQ = ZA.QLS + ZB.QLS
C
C Zone      Ploss Qloss
C (MW)      (MVAR)
C
```

(continues on next page)

(continued from previous page)

```

C NA      $ZA.PLS  $ZA.QLS
C NB      $ZB.PLS  $ZB.QLS
C- - - - -
C Total   $TP      $TQ
C

```

The symbol ZA.PLS contains the losses in MWs for zone NA.

### > DEFINE\_TYPE OWNER\_INDEX

This defines the following records as owner indices. This index is used in conjunction with a coded suffix to obtain specific owner quantities.

Valid suffixes and their associated ownership quantities are shown in the table below.

Table 2.6.24: Suffixes for Ownership Quantities

Suffix	Ownership Quantity
.PG	P_gen in MW
.QG	Q_gen in MVAR
.PL	P_load in MW
.QL	Q_load in MVAR
.PLS	P_loss in MW
.QLS	Q_loss in MW
.PSH	Installed (Scheduled) P_shunt in MW
.QSH	Installed (Scheduled) Q_shunt in MVAR
.SCAP	Installed (Scheduled) Q_cap in MVAR
.SREK	Installed (Scheduled) Q_reactors in MVAR
.UCAP	Used Q_cap in MVAR
.UREK	IUsed Q_reactors in MVAR

The following example illustrates these concepts.

```

> DEFINE_TYPE OWNER_INDEX
LET  ZA = BPA
LET  ZB = PPL
    DEFINE_TYPE FUNCTION
    LET TP = ZA.PLS + ZB.PLS
    LET TQ = ZA.QLS + ZB.QLS
C
C Owner      Ploss Qloss
C (MW)      (MVAR)
C
C BPA      $ZA.PLS  $ZA.QLS
C PPL      $ZB.PLS  $ZB.QLS
C- - - - -
C Total    $TP      $TQ
C

```

The symbol ZA.PLS contains the losses in MWs for owner BPA.

There are three remaining types of indices:

```
> DEFINE_TYPE SYSTEM
> DEFINE_TYPE INTERTIE_INDEX
> DEFINE_TYPE AREA_INDEX
```

These commands define following records as system, intertie, and area indices, respectively. Their use is similar to the bus and zone indices.

### > DEFINE\_TYPE TRANSFER\_INDEX

This defines the following records as transfer indices. This index is used in conjunction with a coded suffix to obtain specific transfer quantities.

Valid suffixes and their associated transfer quantities are shown in the table below.

Table 2.6.25: Suffixes for Branch Quantities

Suffix	Quantity
.RKM	Real part of transfer impedance (p.u.)
.XKM	Imaginary part of transfer impedance (p.u.)
.DVP	Sensitivity $d(V1-V2)/dP$ in kV/MW
.DVQ	Sensitivity $d(V1-V2)/dQ$ in kV/MVAR
.DTP	Sensitivity $d(\angle)/dP$ in degrees/MW

The transfer impedance is the point-to-point impedance between two buses. It would represent the incremental (complex) voltage change due to a 1.0 p.u. current injection into bus1 in conjunction with a -1.0 injection out of bus2. It represents the impedance of the entire network with respect to the two terminal nodes.

The transfer sensitivity is the sensitivity of the voltage or angle difference between two buses with respect to a 1 MW or MVAR change in injection between two buses.

The following example obtains the transfer impedance and transfer sensitivities between COULEE 2 13.8 - JOHN DAY 500 and between JOHN DAY 500 - MALIN 500.

```
> DEFINE_TYPE TRANSFER_INDEX
LET TX1 = COULEE    500 JOHN#DAY  500
LET TX2 = JOHN#DAY 500 MALIN      500
C
C TRANSFER IMPEDANCES          R (P.U.)          X (P.U.)
C
C COULEE    500 JOHN DAY 500 $TX1.RKM/E12.5  $TX1.XKM/E12.5
C JOHN DAY 500 MALIN      500 $TX2.RKM/E12.5  $TX2.XKM/E12.5
C
C TRANSFER SENSITIVITIES      dV/dP (kV/MW)    dV/dQ (kV/MVAR) dT/dP (deg/MW)
C
C COULEE    500 JOHN DAY 500 $TX1.DVP/E12.5  $TX1.DVQ/E12.5  $TX1.DTP/E12.5
C JOHN DAY 500 MALIN      500 $TX2.DVP/E12.5  $TX2.DVQ/E12.5  $TX1.DTP/E12.5
```

## Pagination Specifications

Pagination specifications pertain to headers and subheaders. The Header record is the most important record. It has an H in column 1.

Each user-defined report must begin with a separate header record. The contents of this record become the first subheader. Additional subheaders can be appended to the report.

The header and subheaders are listed at the top of each page on the user-defined analysis report.

Following the header and optional subheader records are 120-character user-formatted comment text, identified with a C in column 1. These records define the user-defined analysis report.

Only columns 3-120 are used. Column 1 (containing the C) is ignored in the report, while column 2 is interpreted as Fortran carriage control:

” ” = single line spacing

“0” = double line spacing

Symbols whose character fields are to be encoded with numerical values computed from the solved case in residence are prefixed with a \$ and suffixed optionally with a format specification.

Examples

C WEST-OF-RIVER FLOW/NORTH= \$T1/F8.0	WEST-OF RIVER FLOW/SOUTH= \$T2/F8.0
C -----	-----
C ELDORADO/LUGO 500. 1 = \$A1/F8.0	PALOVRE/DEVERS 500. 1 = \$A9/F8.0

There are two symbol substitutions in the first comment line: T1 and T2. Both elect an optional format, which is F8.0. By coincidence, the symbol field with \$T1/F8.0 is eight characters, the same as the Format specification. If the Format was larger, substitution would overwrite additional columns on the right. If the format was smaller, only the left-most characters would be used with a blank fill on the remaining (unused) field.

The second comment line has no substitution. The third has two symbol substitutions, similar to the first comment line.

If no format specification is used, the default (F6.0) is used. In this instance, it would be plausible to use five-character symbol names. When the \$ is included, the substituted text is the same field width as the original text. The program limits are 500 comment lines including headers and subheaders. No symbol substitution occurs on headers or subheaders.

If a comment refers to an undefined symbol, a warning is issued, with the questionable fields flagged with a string of ??????`s. The limits are 500 lines of symbol definition and 1000 symbols. Comment text (C in column 1) is excluded.

### 2.6.52 PFC Examples

This section gives several examples of PFC files that can be used with the bpf process.

## Base Case Example

Here is a basic powerflow run that contains all data in the control file (ninebus.pfc).

```
( POWERFLOW,CASEID=NINEBUS, PROJECT = EXAMPLES )
/ HEADER
H WSCC Nine Bus Base Case
/ COMMENT
C
C CASEB-1_ NINE-BUS CASE, ON CARDS, THAT IN ADDITION TO TESTING
C THE FEATURES OF CASE A-1, ALSO FEATURES_TRANSFORMERS
C AND SUBTYPE "E" BUSES.
C
/ P_INPUT_LIST,FULL
/ P_OUTPUT_LIST,FULL
/ AI_LIST=NONE
/ P_ANALYSIS_RPT,LEVEL = 4
/ F_ANALYSIS_RPT,LEVEL = 1
/ NEW_BASE, FILE = ninebus.bse
/ NETWORK_DATA
B GEN1 HI 230 2 -0.0 -0.0 0.0 0.0 -0.0 -0.0 -0.0
B GEN2 HI 230 1 -0.0 -0.0 0.0 0.0 -0.0 -0.0 -0.0
B GEN3 HI 230 2 -0.0 -0.0 0.0 0.0 -0.0 -0.0 -0.0
B STA A 230 1125.0 50.0 0.0 0.0 -0.0 -0.0 -0.0
B STA B 230 2 90. 30.
B STA C 230 2100.0 35.0 0.0 0.0 -0.0 -0.0 -0.0
BS GEN1 16.5 2 -0.0 -0.0 0.0 0.0 71.6 -0.0 -0.01040
BE GEN2 18 1 -0.0 -0.0 0.0 0.0 163.0 -0.0 -0.01025
BE GEN3 13.8 2 -0.0 -0.0 0.0 0.0 85.0 -0.0 -0.01025
L GEN1 HI 230 STA B 230 1 1700 9200 7900
L GEN1 HI 230 STA B 230 2 1700 9200 7900
L GEN1 HI 2302STA A 230 1 85 88
L GEN3 HI 230 STA B 230 39 17 179
L STA C 230 GEN3 HI 230 1190 10080 10450
L STA A 230 GEN2 HI 230 32 161 153
L GEN2 HI 2302STA C 230 85 72 745
T GEN1 HI 230 GEN1 16.5 5760 23000 1650
T GEN2 HI 230 GEN2 18 6250 23000 1800
T GEN3 HI 230 GEN3 13.8 5860 23000 1380
(STOP)
```

A more convenient method to perform the preceding setup is to use a NETWORK\_DATA command and provide the network data portion (the power system network) in a separate file and similarly a INCLUDE\_CONTROL command and provide the control commands in a separate file as well. The PFC file would look like this:

```
( POWERFLOW,CASEID=NINEBUS, PROJECT = EXAMPLES )
/ HEADER
H WSCC Nine Bus Base Case
/ COMMENT
C
C CASEB-1_ NINE-BUS CASE, ON CARDS, THAT IN ADDITION TO TESTING
C THE FEATURES OF CASE A-1, ALSO FEATURES TRANSFORMERS
C AND SUBTYPE "E" BUSES.
C
```

(continues on next page)

(continued from previous page)

```

/ NEW_BASE, FILE= ninebus.bse
/ INCLUDE_CONTROL, FILE = ninebus.ctl
/ NETWORK_DATA, FILE = ninebus.net
(STOP)

```

Where ninebus.ctl contains:

```

/ P_INPUT_LIST, FULL
/ P_OUTPUT_LIST, FULL
/ AI_LIST=NONE
/ P_ANALYSIS_RPT, LEVEL = 4
/ F_ANALYSIS_RPT, LEVEL = 1

```

and ninebus.net contains:

B	GEN1 HI	230	2	-0.0	-0.0	0.0	0.0	-0.0	-0.0	-0.0
B	GEN2 HI	230	1	-0.0	-0.0	0.0	0.0	-0.0	-0.0	-0.0
B	GEN3 HI	230	2	-0.0	-0.0	0.0	0.0	-0.0	-0.0	-0.0
B	STA A	230	1125.0	50.0	0.0	0.0	0.0	-0.0	-0.0	-0.0
B	STA B	230	2	90.	30.					
B	STA C	230	2100.0	35.0	0.0	0.0	0.0	-0.0	-0.0	-0.0
BS	GEN1	16.5	2	-0.0	-0.0	0.0	0.0	71.6	-0.0	-0.01040
BE	GEN2	18	1	-0.0	-0.0	0.0	0.0	163.0	-0.0	-0.01025
BE	GEN3	13.8	2	-0.0	-0.0	0.0	0.0	85.0	-0.0	-0.01025
L	GEN1 HI	230	STA B	230	1	1700	9200			7900
L	GEN1 HI	230	STA B	230	2	1700	9200			7900
L	GEN1 HI	230	2STA A	230	1	85				88
L	GEN3 HI	230	STA B	230	39	17				179
L	STA C	230	GEN3 HI	230	1190	10080	10450			
L	STA A	230	GEN2 HI	230	32	161				153
L	GEN2 HI	230	2STA C	230	85	72				745
T	GEN1 HI	230	GEN1	16.5		5760			23000	1650
T	GEN2 HI	230	GEN2	18		6250			23000	1800
T	GEN3 HI	230	GEN3	13.8		5860			23000	1380

## Change Case Example

Here is an example of loading a system from a solved old base case, and make data changes, and save a new base.:

```

( POWERFLOW, CASEID = TEST-CHG, PROJECT = TEST-WSCC-DATA)
/ NEW_BASE, FILE = 9BUSNEW.BSE
/ COMMENTS
C CASEB-1_ NINE-BUS CASE, ON CARDS, THAT IN ADDITION TO TESTING THE
C FEATURES OF CASE A-1, ALSO FEATURES TRANSFORMERS AND
C SUBTYPE "E" BUSES.
C THE BUS_BRANCH FILE AND THE CHANGE FILE ARE REMOTE
/ INCLUDE_CONTROLS, FILE = TESTCONT.CTL
/ OLD_BASE, FILE= ninebus.bse
/ CHANGES, FILE = CHANG.DAT
( STOP - END OF TEST )

```

**Note:** PFC language commands are not performed in the order they are encountered in the file, but rather in the order the bpf program decides is logical.

---

### Merge Case Example 1

Here is an example of merging two systems defined from separate solved old base files.:

```
( POWERFLOW, CASEID = TEST-MERGE, PROJECT = TEST-MERGE_OLD_BASE )
/COMMENTS
C CASE 2 - TEST BASE MERGE BY MERGING TWO IDENTICAL BASE SYSTEMS.
C       TWO MUTUALLY EXCLUSIVE SUBSYSTEMS ARE INTEGRATED TO
C       REGENERATE THE ORIGINAL SYSTEM.
C
C EACH SYSTEM IS BUILT FROM DIFFERENT AREAS OF THE SAME OLDBASE
.
.   control options
.
/ P_INPUTLIST,FULL
/ F_INPUTLIST,NONE
/ P_OUTPUTLIST,FULL
/ F_OUTPUTLIST,NONE
/ AILIST=FULL
.
/ NEW_BASE, FILE = MERGOLD.BAS
.
.   DEFINE SUBSYSTEM "AREA 1"
.
/ MERGE_OLD_BASE, SUB_SYSTEM_ID = AREA-1, OLD_BASE = TESTDC.BAS
> USE_AIC
> SAVE_AREAS
A AREA 1
.
.   DEFINE SUBSYSTEM "AREA 2"
/ MERGE_OLD_BASE, SUB_SYSTEM_ID = AREA-2,OLD_BASE = TESTDC.BAS
> SAVE_ AREAS
A AREA 2
.
.   SUBSYSTEMS ARE NOW MERGED
.
.   ( CHANGES ) may now follow
.
( STOP )
```



## Merge Case Example 2

Here is an example of merging two topologically overlapping networks into one consolidated network and solving the network, creating a new base to be called J86JFY82. Each of the original networks is to be appropriately modified before the merger. The first network is a WSCC base case saved as 86J201.BSE which must be modified by saving areas, excluding buses, renaming buses and excluding certain branches. The second network is the BPA system which will be extracted from the branch file BDFY82W using the extraction date Jan 1986.:

```
(POWERFLOW, CASEID = J86FY82, PROJECT = BASEMERGE)
/NEWBASE FILE = [APF]J86FY82.BSE
.
.Note:  composite network will be solved with defaults.
.
/MERGE_OLD_BASE,SUBSYSID = WSCC_NETWORK,OLD_BASE=86J201.BSE
>SAVE_AREAS
.....
..... "A" - records - name fields only
.....
>EXCLUDE_BUSES
.....
..... "B" - records - name fields only
.....
>RENAME_BUSES
.....
..... rename table
.....
>EXCLUDE_BRANCHES
.....
..... "L" - records - name fields only
.....
/MERGE_NEW_BASE,SUBSYSID = BPA_NETWORK,BRANCH_DATA=BDFY84,DATE=186
.....
..... "B" - records for BPA system
.....
/CHANGES
.....
..... change records
.....
(STOP)
```

## Reduction Case Example

Here is an example of reducing an existing network to a desired size and solving the reduced network. Reduction is achieved by retaining only specified zones from the original system. Produce full input/output listings on microfiche. Partial input/output listings (restricted to certain specified zones) will be printed on paper. Give full analysis report on both paper and fiche. In solving the network, regulating transformers will be activated and area-interchange control will be switched to control mode. Provide full listing of area interchange flows.:

```
(POWERFLOW, CASEID = A86FY81RED, PROJECT = SAMPLE_PCL)
/OLDBASE, FILE = A8601FY81.BA2
/REDUCTION
>SAVE_ZONES,NA,NB,NC,ND,NE,NF,NG,NH,NI,NJ,NR
```

(continues on next page)

(continued from previous page)

```

>SAVE_ZONES 19,17,20,08,PR,27,16
/P_INPUT_LIST, ZONES=NA,NB,NC,ND,NE,NF,NG,NH,NI,NJ,NK
/P_INPUT_LIST, ZONES = 19,17,20,08,PR
/P_OUTPUT_LIST, ZONES= NA,NB,NC,ND,NE,NF,NG,NH,NI,NJ,NK
/P_OUTPUT_LIST, ZONES= 19,17,20,08,PR
/LTC = ON
/AI_CONTROL = CON
/AI_LIST = FULL
/P_ANALYSIS_RPT, LEVEL = 4
(STOP)

```

## 2.7 Powerflow Command Language (PCL)

### 2.7.1 Introduction

The commands listed below are those currently available in the Powerflow Command Language (PCL). This language was developed to meet the needs of communication between the GUI and the Powerflow server (`ipfsrv`), but is also available for direct user entry via the Command Dialog in the GUI or loading of a PCL command file in the GUI, and via the `ipfbat` command line process, by invoking the same command file. However, since `ipfsrv` is expecting these commands to have been generated by another program, it expects them to be syntactically perfect and legally ordered. There is no checking of input, or confirmation procedure, or chance to edit and re-enter commands. A typo can easily crash the program.

The GUI process sends these commands in response to various button pushes, and all checking, etc. is done before a command is ever sent. The generated text strings are placed in a buffer which is sent over an interprocess communications channel open to the server process (`ipfsrv`). The server process on its end interprets these commands and responds by changing its memory resident case data, querying the case data, sending data back to the GUI, etc. CFLOW library routines also send and receive these commands in the same manner.

PCL has some overlap with the PFC command language used in the batch Powerflow (`bpf`) program. For those commands which do the same things (e.g. `OLD_BASE`, `CHANGES`, etc.) the syntax is identical. However, many of the batch commands are not available in PCL, and of course PCL contains many commands which are not needed by `bpf`. The same conventions on case and spacing apply to both languages.

- Upper, lower, or mixed case are legal: `OLD_BASE`, `Old_Base`, `oldbase`.
- Spaces between command elements are ignored: `/oldbase,file=std.bse` and `/ OLD_base , File = std.bse` are equivalent.
- Underscore may be used within command words if desired: `oldbase`, `old_base`, `O_L_D__B_A_S_E`.

Table 2.7.1: IPC Commands

Command	Description
(END)	Terminates a data stream following a command.
*[EOM]	Used in the Command Dialog to launch a command stream.
^^CFLOW^	Launches a CFLOW program.
CHANGES	Introduces system data change records.
GET_DATA	Fetches data from the Powerow process.
GET_DATA, TYPE = A_DATA	Retrieves all type A input data records in WSCC format.
GET_DATA, TYPE = AREA_DATA	Initializes the user analysis arrays.
GET_DATA, TYPE = AREA_LIST	Loads the area list dialog into the Network Data Edit Dialog and the Report

Table 2.7.1 – continued from

Command	Description
GET_DATA, TYPE = BSEKV_LIST	Loads the base kV list dialog into the Network Data Edit Dialog and Reports Dialog.
GET_DATA, TYPE = BUS_EXISTS	Inquires whether a given bus exists.
GET_DATA, TYPE = BUS_LIST	Loads the bus list dialog in the Alpha Search, Network Data Edit, and Reports Dialog.
GET_DATA, TYPE = BUS_VOLTAGES	Returns a list of all bus voltages.
GET_DATA, TYPE = CONNECTION	Retrieves the network connection diagram for a given set of buses.
GET_DATA, TYPE = COUNT	Computes the number of network data records satisfying the filter criteria.
GET_DATA, TYPE = FILE_EXISTS	Inquires whether a named file exists.
GET_DATA, TYPE = INITIALIZE_DEF	Initializes the user analysis arrays.
GET_DATA, TYPE = INPUT	Retrieves the full network data for a bus.
GET_DATA, TYPE = I_DATA	Retrieves all I type data in WSCC format.
GET_DATA, TYPE = LINE_IMPEDANCE_CALCULATION	Computes transmission line impedance data given tower geometry and conductor data.
GET_DATA, TYPE = LOAD_AREA	Initializes arrays associated with area interchange data.
GET_DATA, TYPE = SUB_DEFINE	Loads the user analysis arrays.
GET_DATA, TYPE = LOAD_REF_AREA	Initializes arrays associated with area interchange data.
GET_DATA, TYPE = LOAD_REF_BASE	Loads a reference base case history file for the purpose of base case comparison.
GET_DATA, TYPE = NETWORK_DATA	Filters network data records.
GET_DATA, TYPE = OUTAGES	Retrieves the list of outaged data for the Report Dialog.
GET_DATA, TYPE = OUTPUT	General purpose command for accessing virtually the entire network data base.
GET_DATA, TYPE = OWNER_LIST	Loads the ownership list dialog into the Reports Dialog and the Network Data Edit Dialog.
GET_DATA, TYPE = RECORD_LIST	Loads the record type list dialog into the Reports Dialog and the Network Data Edit Dialog.
GET_DATA, TYPE = COMMENTS	Obtains the area interchange output data from the WSCC-formatted input and output files.
GET_DATA, TYPE = REF_OUTPUT	General purpose command for accessing virtually the entire network data base.
GET_DATA, TYPE = STATUS	Retrieves the case description.
GET_DATA, TYPE = SUB_DEFINE	Performs character string substitution using computed base case quantities.
GET_DATA, TYPE = COMMENTS	Retrieves all IPF system parameters describing the case in residence.
GET_DATA, TYPE = ZONE_LIST	Loads the zone list dialog into the Reports Dialog and the Network Data Edit Dialog.
INITIALIZE	Starts up powerflow engine.
NETWORK_DATA	Specifies that a network data file is to be loaded into the powerflow engine.
NEW_BASE	Saves the solved, resident base case.
OLD_BASE	Loads a previously solved powerflow case file.
PLOT	Plots a powerflow coordinate file.
PUT_DATA, TYPE = COMMENTS	Defines IPF system parameters which either describe the case in residence or are used for comparison.
QUIT, EXIT	Executes closing procedures and exits the powerflow engine.
REPORTS, SELECT AI_SUMMARY	Retrieves filtered area interchange output data.
REPORTS, SELECT BUS_BR_INPUT	Retrieves filtered WSCC-formatted bus and branch input data records.
REPORTS, SELECT BUS_BR_OUTPUT	Retrieves filtered bus and branch output records.
REPORTS, SELECT BUS_INPUT	Retrieves filtered WSCC-formatted bus input data records.
REPORTS, SELECT BUS_UVOV	Retrieves filtered under/over voltage bus output data.
REPORTS, SELECT LINE_COMPARISON	Retrieves filtered line loading differences between the base case in residence and a comparison case.
REPORTS, SELECT NETWORK_CHANGES	Retrieves the list of all accumulated changes performed on the base case in residence.
REPORTS, SELECT NETWORK_DELETIONS	Retrieves the list of all deleted network data in WSCC format.
REPORTS, SELECT OVERLOADED_LINES	Retrieves filtered overloaded branch output data.
REPORTS, SELECT OVERLOADED_TXS	Retrieves filtered overloaded transformer output data.
REPORTS, SELECT PHASE_SHIFTER	Retrieves the phase shifter report.
REPORTS, SELECT TIE_LINE_SUMMARY	Retrieves filtered area tie line flows.
REPORTS, SELECT VOLTAGE_COMPARISON	Retrieves filtered voltage differences between the resident base case and a comparison case.
SAVE_FILE	Saves the solved resident base case in a named file.
SOLUTION	Causes the powerflow engine to solve the currently resident base case.
SYSCAL	Passes a system command to the operating system.

## 2.7.2 General

### (END)

Use this command to terminate a data stream following a command in a PCL command file. Some commands which would be followed by data are /CHANGES, FILE=\* and /PLOT.

Example:

```
/Network_Data, File = *  
B ARAPAOA115.  
BS ARAP1 13.8 3.5 45. 1.05  
B SHERIDAN115. 41.5 11.8  
T ARAPAOA115. ARAP1 13.8 .00672.14684.00064-00313115.5 13.2  
L ARAPAOA115. SHERIDAN115. .00261.00657 .0004  
(end)
```

### \*[EOM]

This is used when entering commands in the Command Dialog of the GUI, to indicated that the line or lines you have entered into the buffer should be send to ipfsrv. When the GUI generates this “End of Message” string, it has all six characters; however, the first two characters are all that are required to be entered.

For Command Dialog usage, this also serves in place of the (END) command above. But unlike it, this must be entered after every command, when using the Command Dialog.

### INITIALIZE

This command starts up the powerflow engine, and therefore will never be entered by a GUI user, but is required at the beginning of every ipfbat command file. It calls p\_pfini.t.f with the following parameters.:

```
integer function p_pfini (in_buffer, out_buffer)  
parameter (MAXBUFFER = 6600)  
character in_buffer * (MAXBUFFER)  
character out_buffer * (MAXBUFFER)
```

The character array in\_buffer contains the following information.:

```
/INITIALIZE
```

### QUIT, EXIT

This command in either form executes closing procedures and exits the powerflow engine. It calls p\_pfexi.t.f with the following parameters.:

```
integer function p_pfexi (in_buffer, out_buffer)  
parameter (MAXBUFFER = 6600)  
character in_buffer * (MAXBUFFER)  
character out_buffer * (MAXBUFFER)
```

The character array in\_buffer contains either of the following lines of information.:

```
/QUIT
/EXIT
```

### SYSCAL

This command passes a string to the operating system for execution. It is used by the GUI to send a plot to the currently selected print destination, but it will pass any command string.

Example:

```
/SYSCAL
lp -d COMPAQ20 -T ps 0102hw1.ps
(END)
```

**Warning:** This command is effectively a command line injection vulnerability if running separate client and server.

## 2.7.3 File Opening and Saving

### NETWORK\_DATA

This command specifies that a network data file is to be loaded into the ipfsrv. It calls `p_gtnetdat.f` with the following parameters.:

```
integer function p_gtnetdat (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

Example:

```
/NETWORK_DATA, FILE=0102hw1.net
```

For a full description of the command `/NETWORK_DATA`, see the [NETWORK\\_DATA](#). Successful execution should return an IPF state of 2.

### NEW\_BASE

This command saves the solved base case in residence in the named file. It calls `p_newbse.f` with the following parameters.:

```
integer function p_newbse (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains the following information:

```
/NEW_BASE, FILE = <filename>  
C < case comments - three records maximum >  
C < case comments - three records maximum >  
C < case comments - three records maximum >
```

The comment records in the above command are optional. Successful execution should return an IPF state of 6.

### OLD\_BASE

This command specifies that a previously solved power flow case is to be loaded from the specified file and used as the base system for the current request. It calls `p_gtbase.f` with the following parameters.:

```
integer function p_gtbase (in_buffer, out_buffer)  
parameter (MAXBUFFER = 6600)  
character in_buffer * (MAXBUFFER)  
character out_buffer * (MAXBUFFER)
```

Example:

```
/OLD_BASE, FILE = 97hs1a.bse
```

Successful execution should return an IPF state of 6.

### SAVE\_FILE

This command saves the solved base case in residence in the named file in one of four forms. One of these forms is NEW\_BASE, described previously, and admits a redundancy into the command procedure.

This command calls `p_svfile.f` with the following parameters.:

```
integer function p_svfile (in_buffer, out_buffer)  
parameter (MAXBUFFER = 6600)  
character in_buffer * (MAXBUFFER)  
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains any of following information.:

```
/SAVE_FILE, TYPE = SYSTEM_CHANGES, FILE = <filename>  
  
/SAVE_FILE, TYPE = NEW_BASE, FILE = <filename>  
  
/SAVE_FILE, TYPE = NETWORK_DATA, FILE = <filename>, ...  
  
/SAVE_FILE, TYPE = WSCC_BINARY_STABILITY, FILE = <filename>  
                WSCC_ASCII_STABILITY  
  
/SAVE_FILE, TYPE = NETWORK_DATA,  
  FILE = < filename > ,  
  DIALECT = < value > , BPA | WSCC | WSCC1 | PTI  
  RATINGS = < value > , EXTENDED | NOMINAL | MINIMUM  
  SIZE = < value > , 120 | 80
```

See *ipfnet*, for a complete description of the Dialect, Ratings, and Size options when saving a network data file.

## 2.7.4 Processes

### CFLOW

This command launches a CFLOW program if it is in your directory search path. The following works on a VAX VMS system if <program name> is defined as a foreign command::

```
/CFLOW
PROGRAM = my_cflow_program
<program name>::= $dev:[dir]file.exe
```

ARGS = is required only if the CFLOW program requires command line arguments other than the socket number. Any I/O that the CFLOW program does to standard input or standard output will be to and from the same terminal window that the ipfbat program is run from (intermixed with any I/O from the ipfbat program).

### CHANGES

This command introduces system data change records. It calls `p_change.f` with the following parameters.:

```
integer function p_change (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

Examples::

```
/CHANGES, FILE = 0102hw1.chg
/changes, file = *
B D ARAPAHOA115.
BEM ARAP1 13.8 1.05
T M ARAPAHOA 115. ARAP1 13.8 .00672.
(end)
```

### PLOT

This command creates a PostScript diagram (map) file, using a coordinate file and the currently loaded base case. It calls `plot_load.f`, which is the main routine for the batch program `ipfplot`. The batch program takes three parameters, as shown below, and uses the second case, if provided, to produce difference plots.:

```
subroutine plot_load (coord_file, base1_file, base2_file)
character * 60 coord_file, base1_file, base2_file
```

Difference plots are not available from the GUI or ipfbat. The parameters required in the PCL formulation are a coordinate file and the name of a PostScript output file. Records which follow these two file names are interpreted as comments to be placed on the map, following any comments (C records) which occur in the coordinate file. However, if a comment begins with an ampersand (&) or an "at" symbol (@), it will not be printed. The & precedes the name of at most one auxiliary coordinate file to be included on the map. The @ signals the presence of a plot option which will override that option in the coordinate file. See *Network Diagrams* for complete information on plotting diagrams. See *ipfplot* for detailed information on use of the ipfplot program.

Example::

```
/plot
hvmmap.cor
98hw3_study.ps
Transfer Study for PAST
Prepared by A. P. Planner
&transfer.cor
@OP Offset=12.7 18.9
(end)
```

### SOLUTION

This command causes the powerflow engine to solve the currently resident base case. It calls `p_solton.f` with the following parameters.:

```
integer function p_solton (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains the following information. Normal defaults are shown; optional items are in [ ]:

```
/SOLUTION
> BASE_SOLUTION ]
> DEBUG,TX=OFF,BUS=OFF,AI=OFF,DCMODEL=OFF [ON]
> LTC = ON [OFF, ON_NV, ON_NPS, ON_DCONLY]
> AI_CONTROL = CONTROL [MON, OFF]
> MISC_CNTRL, -
    X_BUS = BPA,- [WSCC]
    PHASE_SHIFTER_BIAS = BPA,- [WSCC]
    DCLP = ON, - [OFF]
    VFLATSTART = ON, - [OFF]
    ITER_SUM = OFF, - [ON]
    TSTART = 0.5, -
    NUMVSTEPS = 3
> SOL_ITER, DECOUPLED = 2, NEWTON = 30
> LIMITS,QRES=0.01,PHA=45.0,DEL_ANG=1.0,DEL_VOLT=0.15
> TOLERANCE, BUSV = 0.005, AIPOWER = 0.001, TX = 0.001, Q = 0.005
```

### GET\_DATA

This command with its many different forms fetches data from the powerflow engine. It calls `p_gtdata.f` with the following parameters.:

```
integer function p_gtdata (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains any of the following commands.:



```

/GET_DATA, TYPE = A_DATA
/GET_DATA, TYPE = AREA_DATA
/GET_DATA, TYPE = AREA_LIST
/GET_DATA, TYPE = BSEKV_LIST
/GET_DATA, TYPE = BUS_EXISTS
/GET_DATA, TYPE = BUS_LIST
/GET_DATA, TYPE = BUS_VOLTAGES
/GET_DATA, TYPE = COMMENTS
/GET_DATA, TYPE = CONNECTION
/GET_DATA, TYPE = COUNT
/GET_DATA, TYPE = FILE_EXISTS
/GET_DATA, TYPE = I_DATA
/GET_DATA, TYPE = INITIALIZE_DEF
/GET_DATA, TYPE = INPUT
/GET_DATA, TYPE = LINE_IMPEDANCE_CALCULATION
/GET_DATA, TYPE = LOAD_AREA
/GET_DATA, TYPE = LOAD_DEFINE
/GET_DATA, TYPE = LOAD_REF_AREA
/GET_DATA, TYPE = LOAD_REF_BASE
/GET_DATA, TYPE = NETWORK_DATA
/GET_DATA, TYPE = OUTAGES
/GET_DATA, TYPE = OUTPUT
/GET_DATA, TYPE = OWNER_LIST
/GET_DATA, TYPE = RECORD_LIST
/GET_DATA, TYPE = REF_AREA_DATA
/GET_DATA, TYPE = REF_OUTPUT
/GET_DATA, TYPE = SOL_PAR
/GET_DATA, TYPE = STATUS
/GET_DATA, TYPE = SUB_DEFINE
/GET_DATA, TYPE = SYSTEM
/GET_DATA, TYPE = ZONE_LIST

```

The routine `p_gtdata.f` parses these commands and calls a subroutine to perform the specific task, according to the type of data indicated.

#### **GET\_DATA, TYPE = A\_DATA**

This command retrieves in `out_buffer` all type A input data records in WSCC format. It calls `a_data.f` with the following parameters.:

```

integer function a_data (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)

```

The character array `in_buffer` contains the following information.:

```

/GET_DATA, TYPE = A_DATA

```

Note that no areas are specified in `in_buffer`; this command just gets a list of the areas in the case.:

```

.. _pcl-get-data-area-data:

```

### GET\_DATA, TYPE = AREA\_DATA

This command obtains the area interchange output data from the WSCC-formatted input area records. This command should be preceded with a prior command GET\_AREA, TYPE=LOAD\_AREA. It calls p\_gtbase.f with the following parameters.:

```
integer function p_gtdata (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array in\_buffer contains the following information.:

```
/GET_DATA, TYPE = AREA_DATA
A <areaname>
```

### GET\_DATA, TYPE = AREA\_LIST

This command loads the area list filter window in the Network Data Edit Dialog and the Report Dialog of the GUI. It returns in out\_buffer the list of area names in the following format.:

```
<areaname> LINEFEED
```

where <areaname> is the area name in A10 format.

It calls area\_list.f with the following parameters.:

```
integer function area_list (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array in\_buffer contains optional filter data in the following command.:

```
/GET_DATA, TYPE = AREA_LIST [ FROM BUS_DATA ]
      WHERE AREAS = <area1>, <area2>, etc AND
      ZONES = <zone1>, <zone2>, etc AND
      OWNERS = <own1>, <own2>, etc AND
      BASEKV = base1
              < base ( example < 115.0 means all base
                  kv's less than or equal to 115.0)
              > base ( example > 115.0 means all base
                  kv's greater than or equal to 115.0)
              base1 < base2 (all bases between base1 and
                  base 2)
              base2 > base1 (same as above)
      TYPE = '*' ', 'A*', 'A?', 'I ', 'B*', 'L*', 'B?',
              'B ', 'BE', 'BS', 'BC', 'BD', 'BV', 'BQ',
              'BG', 'BT', 'BX', 'BM', 'BF', '+ ', 'X ',
              'Q ', 'LD', 'LM', 'E ', 'T ', 'TP', 'R ',
              'RZ'
      ALL
```

Details of the filter are found in *Dynamic Filters*.

**GET\_DATA, TYPE = BSEKV\_LIST**

This command loads the base kV filter window in the Network Data Edit Dialog and the Reports Dialog of the GUI. It returns in out\_buffer the list of filtered base kVs in the following format.:

```
<basekv> LINEFEED
```

where <basekv> is the base kV in F6.1 format.

It calls bsekvlst.f with the following parameters.:

```
integer function bsekvlst (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array in\_buffer contains optional filter data in the following command.:

```
/GET_DATA, TYPE = BSEKV_LIST [ FROM BUS_DATA ]
      WHERE AREAS = <area1>, <area2>, etc AND
      ZONES = <zone1>, <zone2>, etc AND
      OWNERS = <own1>, <own2>, etc AND
      BASEKV = base1
              < base ( example < 115.0 means all base
                    kv's less than or equal to 115.0)
              > base ( example > 115.0 means all base
                    kv's greater than or equal to 115.0)
      base1 < base2 (all bases between base1 and
                    base 2)
      base2 > base1 (same as above)
      TYPE = '* ', 'A*', 'A?', 'I ', 'B*', 'L*', 'B?',
            'B ', 'BE', 'BS', 'BC', 'BD', 'BV', 'BQ',
            'BG', 'BT', 'BX', 'BM', 'BF', '+ ', 'X ',
            'Q ', 'LD', 'LM', 'E ', 'T ', 'TP', 'R ',
            'RZ'
      BUS = "<busname>" (quotes are necessary)
      ALL
      LOADING = (<min> <max>)
```

Details of the filter are found in *Dynamic Filters*.

**GET\_DATA, TYPE = BUS\_EXISTS**

This command inquires whether a given bus exists. It calls ex\_bus.f with the following parameters.:

```
integer function ex_bus (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array in\_buffer contains the following information.:

```
/GET_DATA, TYPE = BUS_EXISTS, BUS = bus_name base_kv
```

bus\_name is an eight character name (blank filled to eight characters) followed by a Fortran F4.0 or F6.1 base kV.

The return status is 0 if the bus exists in the current case or 1 if it does not exist.

### GET\_DATA, TYPE = BUS\_LIST

This command loads the bus list dialog in the Alpha Search Dialog, the Network Data Edit Dialog, and the Reports Dialog of the GUI. It returns in out\_buffer the list of filtered bus names and base kVs in the following format.:

```
<busname><base kv> LINEFEED
```

where <busname> is the bus name in A8 format; <basekv> is the base kV in F6.1 format.

It calls bus\_list.f with the following parameters.:

```
integer function bus_list (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array in\_buffer contains optional filter data in the following command.:

```
/GET_DATA, TYPE = BUS_LIST [ FROM BUS_DATA ]
    WHERE AREAS = <area1>, <area2>, etc AND
    ZONES = <zone1>, <zone2>, etc AND
    OWNERS = <own1>, <own2>, etc AND
    BASEKV = base1
        < base ( example < 115.0 means all base
            kv's less than or equal to 115.0)
        > base ( example > 115.0 means all base
            kv's greater than or equal to 115.0)
        base1 < base2 (all bases between base1 and
            base 2)
        base2 > base1 (same as above)
    TYPE = '*' , 'A*' , 'A?' , 'I' , 'B*' , 'L*' , 'B?' ,
        'B' , 'BE' , 'BS' , 'BC' , 'BD' , 'BV' , 'BQ' ,
        'BG' , 'BT' , 'BX' , 'BM' , 'BF' , '+' , 'X' ,
        'Q' , 'LD' , 'LM' , 'E' , 'T' , 'TP' , 'R' ,
        'RZ'
    BUS = "<busname>" (quotes are necessary)
    AFTER_BUS = "<busname>"
    ALL
    LOADING = (<min> <max>)
```

Details of the filter are found in *Dynamic Filters*.

**GET\_DATA, TYPE = BUS\_VOLTAGES**

This command returns in out\_buffer the list of all bus voltages in the following format.:

```
<busname><base kv><voltage><angle> LINEFEED
```

where <busname> **is** the bus name **in** (A8),  
 <basekv> **is** the base kV **in** (F4.0),  
 <voltage> **is** the actual voltage magnitude **in** kV **in** (I4), **and**  
 <angle> **is** the angle **in** degrees **in** (I4).

It calls gtbsvolt.f with the following parameters.:

```
integer function gtbsvolt (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array in\_buffer contains the following information.:

```
/GET_DATA, TYPE = BUS_VOLTAGES
```

**GET\_DATA, TYPE = COMMENTS**

This command obtains case comments, along with case ID, project name, and headers.

There is a related command:

```
/PUT_DATA, TYPE = COMMENTS
```

which modifies the corresponding data except for header 1 which is not modifiable. Header 1 is formatted to include case name, case description, program version, date, etc. Up to 20 comments are returned. The three header records are always returned.

The returned values are encoded in the character array out\_buffer in free field, C-formatted strings. The quantities enclosed in angle brackets "< ... >" denote variables returned. Headers may be up to 130 characters; comments may be up to 120 characters, not including the H or C in column 1.:

```
/GET_DATA, TYPE = COMMENTS
CASE_ID = "< case name >"           10 chars
CASE_DS = "< case description >"      20 chars
H< header 1 information >
H< header 2 information >
H< header 3 information >
C< comment text >
...
C< comment text >
return status: status = 0 : success
                  1 : error
```

### GET\_DATA, TYPE = CONNECTION

This command retrieves network connection information for given buses. Its main usage is to draw the display network diagram. It calls `gtconnect.f` with the following parameters.:

```
integer function gtconnect (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains the following information.:

```
/GET_DATA, TYPE = CONNECTION
B      <busname, etc> returns all connection data associated with bus
B      <busname, etc> returns all connection data associated with bus
```

The connection data is returned in `out_buffer` in the following format.:

```
B <busname><basekv> LINEFEED
L <busname><basekv> <busname><basekv> LINEFEED
T <busname><basekv> <busname><basekv> LINEFEED
```

Specifically.:

```
"B" records: (1:2) = "B "
              (3:14) = <bus1><base1>.
"L" records: (1:2) = "L "
              (3:14) = <bus1><basekv1>
              (16:27) = <bus2><base2>.
              (28:29) = <number of parallel circuits>
"T" records: (1:2) = "T "
              (3:14) = <bus1><basekv1>
              (16:27) = <bus2><base2>.
              (28:29) = <number of parallel circuits>
```

### GET\_DATA, TYPE = COUNT

This command computes the number of network data records that would be retrieved using a subsequent `/GET_DATA, TYPE=NETWORK_DATA` command using the same filter which is defined with this command. The output appears in a dialog field in the Network Data Edit Dialog of the GUI. It returns the count in `out_buffer` in the following format.:

```
count = ccccc
```

cccccc is the count in I6 format. This command calls `gtcount.f` with the following parameters.:

```
integer function gtcount (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains filter data in the following command.:

```

/GET_DATA, TYPE = COUNT [ FROM BUS_DATA ]
WHERE AREAS = <area1>, <area2>, etc AND
ZONES = <zone1>, <zone2>, etc AND
OWNERS = <own1>, <own2>, etc AND
BASEKV = base1
    < base ( example < 115.0 means all base
        kv's less than or equal to 115.0)
    > base ( example > 115.0 means all base
        kv's greater than or equal to 115.0)
    base1 < base2 (all bases between base1 and
        base 2)
    base2 > base1 (same as above)
TYPE = '*' , 'A*' , 'A?' , 'I' , 'B*' , 'L*' , 'B?' ,
        'B' , 'BE' , 'BS' , 'BC' , 'BD' , 'BV' , 'BQ' ,
        'BG' , 'BT' , 'BX' , 'BM' , 'BF' , '+' , 'X' ,
        'Q' , 'LD' , 'LM' , 'E' , 'T' , 'TP' , 'R' ,
        'RZ'
BUS = "<busname>" (quotes are necessary)
AFTER_BUS = "<busname>"
ALL
LOADING = (<min> <max>)

```

### GET\_DATA, TYPE = FILE\_EXISTS

This command inquires whether a named file exists on the platform where ipvsrv is running. It calls `ex_file.f` with the following parameters.:

```

integer function ex_file (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)

```

The character array `in_buffer` contains the following information.:

```

/GET_DATA, TYPE = FILE_EXISTS, FILE = <file_name>

```

The return status is 0 if the bus exists, or 1 if it does not exist.

### GET\_DATA, TYPE = I\_DATA

This command retrieves in `out_buffer` all type I records in WSCC format. It calls `i_data.f` with the following parameters.:

```

integer function i_data (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)

```

The character array `in_buffer` contains the following information:

```

/GET_DATA, TYPE = I_DATA

```

### GET\_DATA, TYPE = INITIALIZE\_DEF

This command initializes the user analysis arrays. It should be called prior to a USER\_ANALYSIS command. It calls `p_initdef.f` with the following parameters.:

```
integer function p_initdef (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains the following information.:

```
/GET_DATA, TYPE = INITIALIZE_DEF
```

### GET\_DATA, TYPE = INPUT

This command retrieves the full network data given the identification of that record in WSCC format. If the record is type B, all data relevant to that bus is retrieved. It calls `gtinput.f` with the following parameters.:

```
integer function gtinput (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains WSCC-formatted records following the command `/GET_DATA.:`

```
/GET_DATA, TYPE = INPUT
A <areaname>
I <area1 area2>
B <busname, etc> returns all data associated with bus
+ <busname, etc> returns all data if id fields have wild cards
    (type - column 2, owner, columns 3-5, and code-year
    columns 20-21)
X <busname, etc>
L <bus1 bus2, etc> returns all parallels if id is wild card (*)
    returns all sections if section is 0
T <bus1 bus2, etc>
R <bus1 bus2, etc>
E <bus1 bus2, etc>
```

The character array `in_buffer` is passed through `p_gtdata.f` to `gtinput.f`.

### GET\_DATA, TYPE = LINE\_IMPEDANCE\_CALCULATION

This command computes transmission line impedance data given tower geometry and conductor characteristics. It calls a stand-alone module `p_lic.f` and associated routines, which are completely separate from the powerflow data base. (It was added to `ipfsrv` to keep the GUI free from any FORTRAN modules.):

```
integer function p_lic (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```



The character array `in_buffer` contains the following information:

```
/GET_DATA, TYPE = LINE_IMPEDANCE_CALCULATION
      UNITS = < ENGLISH | METRIC >,
      DISTANCE = < miles | km >
      BASEKV = <basekv>,
      BASEMVA = <basemva>,
      FREQUENCY = <freq>
CONDUCTOR = 1 .3636 .05215 1.602 -20.75 50. 50. 0.0 0.0 0
CONDUCTOR = 1 .3636 .05215 1.602 -19.25 50. 50. 0.0 0.0 0
CONDUCTOR = 2 .3636 .05215 1.602 -0.75 77.5 77.5 0.0 0.0 0
CONDUCTOR = 2 .3636 .05215 1.602 0.75 77.5 77.5 0.0 0.0 0
CONDUCTOR = 3 .3636 .05215 1.602 19.25 50. 50. 0.0 0.0 0
CONDUCTOR = 3 .3636 .05215 1.602 20.75 50. 50. 0.0 0.0 0
CONDUCTOR = 0 .5 2.61 0.386 -12.9 98.5 98.5 0.0 0.0 0
CONDUCTOR = 0 .5 2.61 0.386 12.9 98.5 98.5 0.0 0.0 0
```

The returned line impedance data in character array `out_buffer` has the following format.:

```
LIC = <r> <x> <g/2> <b/2>
```

`<r>`, `<x>`, `<g/2>`, and `<b/2>` are the corresponding per unit line quantities encoded as F14.8 fields.

### GET\_DATA, TYPE = LOAD\_AREA

This command initializes arrays associated with area interchange data. It should be called prior to any requests for area output. It calls `ldardata.f` with the following parameters.:

```
integer function ldardata (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains the following information.:

```
/GET_DATA, TYPE = LOAD_AREA
```

### GET\_DATA, TYPE = LOAD\_DEFINE

This command loads the user analysis arrays. It calls `p_loaddef.f` with the following parameters.:

```
integer function p_loaddef (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains the following information.:

```
/GET_DATA, TYPE = LOAD_DEFINE
> DEFINE ...
> DEFINE ...
> DEFINE ...
```

(continues on next page)

(continued from previous page)

```
C ...  
C ...  
C ...
```

### GET\_DATA, TYPE = LOAD\_REF\_AREA

This command initializes arrays associated with area interchange data using the data from the reference base case. Prior to this call, the reference base case should be loaded. It calls `P_ldxardta.f` with the following parameters.:

```
integer function p_ldxardta (in_buffer, out_buffer)  
parameter (MAXBUFFER = 6600)  
character in_buffer * (MAXBUFFER)  
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains the following information.:

```
/GET_DATA, TYPE = LOAD_REF_AREA
```

### GET\_DATA, TYPE = LOAD\_REF\_BASE

This command loads a reference base case history file for the purpose of base case comparison or plot comparisons. The requested base case must be in the IPF format. It calls `p_gtdata.f` with the following parameters.:

```
integer function p_gtdata (in_buffer, out_buffer)  
parameter (MAXBUFFER = 6600)  
character in_buffer * (MAXBUFFER)  
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains the following information:

```
/GET_DATA, TYPE = LOAD_REF_BASE, FILE = <file_name>
```

### GET\_DATA, TYPE = NETWORK\_DATA

This command gets filtered network data records. The output appears in the scrollable edit list on the Network Data Edit Dialog in the GUI. This command calls `gtnetdat.f` with the following parameters.:

```
integer function gtnetdat (in_buffer, out_buffer)  
parameter (MAXBUFFER = 6600)  
character in_buffer * (MAXBUFFER)  
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains filter data in the following command.:

```
/GET_DATA, TYPE = NETWORK_DATA [ FROM BUS_DATA ]  
      WHERE AREAS = <area1>, <area2>, etc AND  
            ZONES = <zone1>, <zone2>, etc AND  
            OWNERS = <own1>, <own2>, etc AND  
            BASEKV = base1
```

(continues on next page)

(continued from previous page)

```

< base ( example < 115.0 means all base
kv's less than or equal to 115.0)
> base ( example > 115.0 means all base
kv's greater than or equal to 115.0)
base1 < base2 (all bases between base1 and
base 2)
base2 > base1 (same as above)
TYPE = '*' , 'A*' , 'A?' , 'I' , 'B*' , 'L*' , 'B?' ,
'B' , 'BE' , 'BS' , 'BC' , 'BD' , 'BV' , 'BQ' ,
'BG' , 'BT' , 'BX' , 'BM' , 'BF' , '+' , 'X' ,
'Q' , 'LD' , 'LM' , 'E' , 'T' , 'TP' , 'R' ,
'RZ'
BUS = "<busname>" (quotes are necessary)
AFTER_BUS = "<busname>"
ALL
LOADING = (<min> <max>)

```

Details of the filter are found in *Dynamic Filters*.

### GET\_DATA, TYPE = OUTAGES

This command retrieves the list of outaged data for the Report Dialog (under Bone Pile). It calls gtoutage.f with the following parameters.:

```

integer function gtoutage (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)

```

The character array in\_buffer contains the following information.:

```
/GET_DATA, TYPE = OUTAGES
```

The returned outage data returned in character array out\_buffer is WSCC-formatted network data that has been deleted.

### GET\_DATA, TYPE = OUTPUT

This command can access virtually the entire network data base. It calls gtoutput.f with the following parameters.:

```

integer function gtoutput (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)

```

The character array in\_buffer contains the following information.:

```

/GET_DATA, TYPE = OUTPUT
A <areaname> returns all data associated with area
I <area1 area2> returns all intertie data between the areas
B <busname, etc> returns all data associated with bus
+ <busname, etc> returns all data if id fields have wild cards

```

(continues on next page)

(continued from previous page)

```

                (type - column 2, owner, columns 3-5, and code-year
                columns 20-21)
X <busname, etc>
L <bus1 bus2, etc> returns all parallels if id is wild card (*)
                    returns all sections if section is wild card (0)
T <bus1 bus2, etc>
R <bus1 bus2, etc>
E <bus1 bus2, etc>

```

The returned values in out\_buffer correspond with the input record in in\_buffer.

Table 2.7.2: Area A Record

Column	Format	Description
1	A1	A — Area Identifier
2	1X	Blank
3-12	A10	Area name
13	1X	Blank
14-28	E15.7	Total Area Generation (MW)
29-43	E15.7	Total Area Load (MW)
44-58	E15.7	Total Area Losses (MW)
59-73	E15.7	Net Area Export (MW)

Table 2.7.3: AC Buses (Types B, BE, BS, BC, BD, BV, BQ, BG, BT, BX)

Column	Format	Description
1	A1	Bus code “B”
2	A1	Bus type (“”, “E”, “S”, “D”, etc.)
3	1X	(Not used)
4-6	A3	Ownership
7-14	A8	Bus name
15-18	F4.0	Bus base KV
19-20	A2	Zone
21-35	E15.7	P_gen (MW)
36-50	E15.7	Q_gen (MVAR)
51-65	E15.7	Voltage (KV)
66-80	E15.7	Angle (degrees)
81-95	E15.7	P_load (MW)
96-110	E15.7	Q_load (MVAR)
111-125	E15.7	B_shunt used (MVAR)
126-140	E15.7	B_shunt scheduled (MVAR)
141-155	E15.7	B_shunt (capacitors) used (MVAR)
156-170	E15.7	B_shunt (capacitors) scheduled (MVAR)
171-185	E15.7	B_shunt (reactors) used (MVAR)
186-200	E15.7	B_shunt (reactors) scheduled (MVAR)
201-215	E15.7	Q unscheduled (MVAR)

Table 2.7.4: DC Buses (Type BD and BM)

Column	Format	Description
1	A1	Bus code “B”
2	A1	Bus type (“D” or “M”)
3	1X	(Not used)
4-6	A3	Ownership
7-14	A8	Bus name
15-18	F4.0	Bus base KV
19-20	A2	Zone
21-35	E15.7	P <sub>d-c</sub> (MW)
36-50	E15.7	Q <sub>d-c</sub> (MVAR)
51-65	E15.7	D <sub>C</sub> Voltage (KV)
66-80	E15.7	Converter angle (degrees)
81-95	E15.7	P <sub>valve</sub> losses (MW)
96-110	E15.7	Q <sub>valve</sub> losses (MVAR)

2-6 5X (Not used) 4-6 A3 Ownership 7-14 A8 Bus name 15-18 F4.0 Bus base KV 19-20 2X (not used) 21 I1 Group No. 1 scheduled units 22 I1 Group No. 1 used units 23-37 E15.7 Group No. 1 reactance (MVAR) / unit 38 I1 Group No. 2 scheduled units 39 I1 Group No. 2 used units 40-54 E15.7 Group No. 2 reactance (MVAR) / unit 55 I1 Group No. 3 scheduled units 56 I1 Group No. 3 used units 57-71 E15.7 Group No.3 reactance (MVAR) / unit 72 I1 Group No. 4 scheduled units 73 I1 Group No. 4 used units 74-88 E15.7 Group No. 4 reactance (MVAR) / unit 89 I1 Group No. 5 scheduled units 90 I1 Group No.5 used units 91-105 E15.7 Group No. 5 reactance (MVAR) / unit 106 I1 Group No. 6 scheduled units 107 I1 Group No. 6 used units 108-122 E15.7 Group No. 6 reactance (MVAR) / unit 123 I1 Group No. 7 scheduled units 124 I1 Group No. 7 scheduled units 125-139 E15.7 Group No. 7 reactance (MVAR) / unit 140 I1 Group No. 8 scheduled units 141 I1 Group No. 8 used units 142-156 E15.7 Group No. 8 reactance (MVAR) / unit

2 A1 Continuation bus subtype (A,C,F,I,N,P,S) 3 1X (Not used) 4-6 A3 Ownership 7-14 A8 Bus name 15-18 F4.0 Bus base KV 19-20 A2 Classification code year, \* I — constant current loads, \* Z — constant impedance loads, \* P — constant MVA loads, 21-35 E15.7 P<sub>gen</sub> (MW) 36-50 E15.7 Q<sub>gen</sub> (MVAR) 51-65 E15.7 P<sub>load</sub> (MW) 66-80 E15.7 Q<sub>load</sub>(MVAR) 81-95 E15.7 G<sub>shunt</sub> (MW) 96-110 E15.7 B<sub>shunt</sub> (MVAR)

2 A1 Line subtype (LD, LM, or TP) 3 1X (Not used) 4-6 A3 Ownership 7-14 A8 Bus1 name 15-18 F4.0 Bus1 base KV 19 I1 Interchange metering point (0, 1, or 2) 20-27 A8 Bus 2 name 28-31 F4.0 Bus 2 base KV 32 A1 Parallel ID ( \* (asterisk) means all parallels) 33 I1 Number of circuits 34-48 E15.7 P<sub>in</sub> (MW) 49-63 E15.7 Q<sub>in</sub> (MVAR) 64-78 E15.7 P<sub>out</sub> (MW) 79-93 E15.7 Q<sub>out</sub> (MVAR) 94-108 E15.7 P<sub>loss</sub> (MW) 109-123 E15.7 Q<sub>loss</sub> (MW) 124-138 E15.7 Critical line loading (amps) 139-146 F8.1 Critical line rating (amps) 147 A1 Critical line rating code (N,T,B) 148 I1 Critical line loading terminal (0,1,2) 149-163 E15.7 Critical transformer loading (MVA) 164-171 F8.1 Critical transformer rating (MVA) 172 A1 Critical transformer rating code (N,T,E,B) 173 I1 Critical transformer loading terminal (0,1,2) 174-188 E15.7 Total Line loading (percent) 189-203 E15.7 Total Line loading (amps) 204-218 E15.7 Total Transformer loading (percent) 219-233 E15.7 Total Transformer loading (MVA) 234-241 F8.2 Tap1 in kV (Type T or TP) or %Compensation (L or E.) 242-249 F8.2 Tap2 in kV 250-256 A7 (Reserved for difference plotting)

**GET\_DATA, TYPE = OWNER\_LIST**

This command loads the ownership list dialog in the Reports Dialog and in the Network Data Edit Dialog of the GUI. It returns in out\_buffer the list of filtered ownership names in the following format. <ownership>LINEFEED <ownership> is the ownership name in A3 format. It calls owner\_list.f with the following parameters. integer function owner\_list (in\_buffer, out\_buffer) parameter (MAXBUFFER = 6600) character in\_buffer \* (MAXBUFFER) character out\_buffer \* (MAXBUFFER) The character array in\_buffer contains optional filter data in the following command. /GET\_DATA, TYPE = OWNER\_LIST [ FROM BUS\_DATA ]

WHERE AREAS = <area1>, <area2>, etc AND ZONES = <zone1>, <zone2>, etc AND OWNERS = <own1>, <own2>, etc AND BASEKV = base1 < base ( example < 115.0 means all base kv's less than or equal to 115.0) > base ( example > 115.0 means all base kv's greater than or equal to 115.0) base1 < base2 (all bases between base1 and base 2) base2 > base1 (same as above) TYPE = '\*' , 'A\*', 'A?', 'I' , 'B\*', 'L\*', 'B?',

'B' , 'BE', 'BS', 'BC', 'BD', 'BV', 'BQ',  
'BG', 'BT', 'BX', 'BM', 'BF', '+', 'X', 'Q', 'LD', 'LM', 'E', 'T', 'TP', 'R', 'RZ' BUS = "<busname>" (quotes are necessary) AFTER\_BUS = "<busname>" ALL LOADING = (<min> <max>)

Details of the filter are found in *Dynamic Filters*.

**GET\_DATA, TYPE = RECORD\_LIST**

This command loads the record type list dialog in the Reports Dialog and in the Network Data Edit Dialog of the GUI. It returns in out\_buffer the hard-coded list of various record type codes in the following format. <record\_type>LINEFEED <record\_type> is the record type code in A2 format. This command calls type\_list.f with the following parameters. integer function type\_list (in\_buffer, out\_buffer) parameter (MAXBUFFER = 6600) character in\_buffer \* (MAXBUFFER) character out\_buffer \* (MAXBUFFER) The character array in\_buffer contains optional filter data in the following command. /GET\_DATA, TYPE = RECORD\_LIST [ FROM BUS\_DATA ]

WHERE AREAS = <area1>, <area2>, etc AND ZONES = <zone1>, <zone2>, etc AND OWNERS = <own1>, <own2>, etc AND BASEKV = base1 < base ( example < 115.0 means all base kv's less than or equal to 115.0) > base ( example > 115.0 means all base kv's greater than or equal to 115.0) base1 < base2 (all bases between base1 and base 2) base2 > base1 (same as above) TYPE = '\*' , 'A\*', 'A?', 'I' , 'B\*', 'L\*', 'B?',

'B' , 'BE', 'BS', 'BC', 'BD', 'BV', 'BQ',  
'BG', 'BT', 'BX', 'BM', 'BF', '+', 'X', 'Q', 'LD', 'LM', 'E', 'T', 'TP', 'R', 'RZ' BUS = "<busname>" (quotes are necessary) AFTER\_BUS = "<busname>" ALL LOADING = (<min> <max>)

In this instance only, the filter has no impact upon the contents of the returned data. Details of the filter are found in the IPF Basic User's Guide under the section on "Dynamic Filters." GET\_DATA, TYPE = REF\_AREA\_DATA This command obtains the area interchange output data from the WSCC-formatted input area records using the reference base case data. This command should be preceded with a prior command GET\_AREA, TYPE=LOAD\_REF\_AREA. It calls p\_gtxardta.f with the following parameters. integer function p\_gtxardta (in\_buffer, out\_buffer) parameter (MAXBUFFER = 6600) character in\_buffer \* (MAXBUFFER) character out\_buffer \* (MAXBUFFER) The character array in\_buffer contains the following information. /GET\_DATA, TYPE = AREA\_REF\_DATA A <areaname>

GET\_DATA, TYPE = REF\_OUTPUT This command can access virtually the entire network data from the reference base. It calls gtolpt.f with the following parameters. integer function gtolpt (in\_buffer, out\_buffer) parameter (MAXBUFFER = 6600) character in\_buffer \* (MAXBUFFER) character out\_buffer \* (MAXBUFFER) The character array in\_buffer contains the following information. /GET\_DATA, TYPE = OUTPUT A <areaname> returns all data associated with area I <area1 area2> returns all interties between the two areas B <busname, etc> returns all data associated with bus + <busname, etc> returns all data if id fields have wild cards

(type - column 2, owner, columns 3-5, and code-year columns 20-21)

X <busname, etc> L <bus1 bus2, etc> returns all parallels if id is wild card (\*)

returns all sections if section is wild card (0)

T <bus1 bus2, etc> R <bus1 bus2, etc> E <bus1 bus2, etc> The returned values in out\_buffer correspond with the input record in in\_buffer.

2 1X Blank 3-12 A10 Area name 13 1X Blank 14-28 E15.7 Total Area Generation (MW) 29-43 E15.7 Total Area Load (MW) 44-58 E15.7 Total Area Losses (MW) 59-73 E15.7 Net Area Export (MW)

2 1X Blank 3-12 A10 Area 1 name 13 1X Blank 14-23 A10 Area 2 name 24 1X Blank 25-39 E15.7 Scheduled Area1-Area2 Export (MW) 40-54 E15.7 Actual Area1-Area2 Export (MW) 55-69 E15.7 "Circulating" Flow (MW) 70 1X Blank 71 I1 0 — No Area1-Area2 I record exists 1 — Area1-Area2 I record exists

2 A1 Bus type 3 1X (Not used) 4-6 A3 Ownership 7-14 A8 Bus name 15-18 F4.0 Bus base KV 19-20 A2 Zone 21-35 E15.7 P\_gen (MW) 36-50 E15.7 Q\_gen (MVAR) 51-65 E15.7 Voltage (KV) 66-80 E15.7 Angle (degrees) 81-95 E15.7 P\_load (MW) 96-110 E15.7 Q\_load (MVAR) 111-125 E15.7 B\_shunt used (MVAR) 126-140 E15.7 B\_shunt scheduled (MVAR) 141-155 E15.7 B\_shunt (capacitors) used (MVAR) 156-170 E15.7 B\_shunt (capacitors) scheduled (MVAR) 171-185 E15.7 B\_shunt (reactors) used (MVAR) 186-200 E15.7 B\_shunt (reactors) scheduled (MVAR) 201-215 E15.7 Q unscheduled (MVAR)

2 A1 Bus type ("D" or "M") 3 1X (Not used) 4-6 A3 Ownership 7-14 A8 Bus name 15-18 F4.0 Bus base KV 19-20 A2 Zone 21-35 E15.7 P\_d-c (MW) 36-50 E15.7 Q\_d-c (MVAR) 51-65 E15.7 D\_C Voltage (KV) 66-80 E15.7 Converter angle (degrees) 81-95 E15.7 P\_valve losses (MW) 96-110 E15.7 Q\_valve losses (MVAR)

2-6 5X (Not used) 4-6 A3 Ownership 7-14 A8 Bus name 15-18 F4.0 Bus base kV 19-20 2X (not used) 21 I1 Group No. 1 scheduled units 22 I1 Group No. 1 used units 23-37 E15.7 Group No. 1 reactance (MVAR) / unit 38 I1 Group No. 2 scheduled units 39 I1 Group No. 2 used units 40-54 E15.7 Group No. 2 reactance (MVAR) / unit 55 I1 Group No. 3 scheduled units 56 I1 Group No. 3 used units 57-71 E15.7 Group No.3 reactance (MVAR) / unit 72 I1 Group No. 4 scheduled units 73 I1 Group No. 4 used units 74-88 E15.7 Group No. 4 reactance (MVAR) / unit 89 I1 Group No. 5 scheduled units 90 I1 Group No.5 used units 91-105 E15.7 Group No. 5 reactance (MVAR) / unit 106 I1 Group No. 6 scheduled units 107 I1 Group No. 6 used units 108-122 E15.7 Group No. 6 reactance (MVAR) / unit 123 I1 Group No. 7 scheduled units 124 I1 Group No. 7 scheduled units 125-139 E15.7 Group No. 7 reactance (MVAR) / unit 91 I1 Group No. 8 scheduled units 92 I1 Group No. 8 used units 93-100 E15.7 Group No. 8 reactance (MVAR) / unit

2 A1 Continuation bus subtype (A,C,F,I,N,P,S) 3 1X (Not used) 4-6 A3 Ownership 7-14 A8 Bus name 15-18 F4.0 Bus base KV 19-20 A2 Classification code year, \*I — constant current loads, \*Z — constant impedance loads, \*P — constant MVA loads, 21-35 E15.7 P\_gen (MW) 36-50 E15.7 Q\_gen (MVAR) 51-65 E15.7 P\_load (MW) 66-80 E15.7 Q\_load(MVAR) 81-95 E15.7 G\_shunt (MW) 96-110 E15.7 B\_shunt (MVAR)

2 A1 Line subtype (LD, LM, or TP) 3 1X (Not used) 4-6 A3 Ownership 7-14 A8 Bus1 name 15-18 F4.0 Bus1 base KV 19 I1 Interchange metering point (0, 1, or 2) 20-27 A8 Bus 2 name 28-31 F4.0 Bus 2 base KV 32 A1 Parallel ID ( \* (asterisk) means all parallels) 33 I1 Number of circuits 34-48 E15.7 P\_in (MW) 49-63 E15.7 Q\_in (MVAR) 64-78 E15.7 P\_out (MW) 79-93 E15.7 Q\_out (MVAR) 94-108 E15.7 P\_loss (MW) 109-123 E15.7 Q\_loss (MW) 124-138 E15.7 Critical line loading (amps) 139-146 F8.1 Critical line rating (amps) 147 A1 Critical line rating code (N,T,B) 148 I1 Critical line loading terminal (0,1,2) 149-163 E15.7 Critical transformer loading (MVA) 164-171 F8.1 Critical transformer rating (MVA) 172 A1 Critical transformer rating code (N,T,E,B) 173 I1 Critical transformer loading terminal (0,1,2) 174-188 E15.7 Total Line loading (percent) 189-203 E15.7 Total Line loading (amps) 204-218 E15.7 Total Transformer loading (percent) 219-233 E15.7 Total Transformer loading (MVA) 234-241 F8.2 Tap1 in kV (Type T or TP) or %Compensation (L or E.) 242-249 F8.2 Tap2 in kV 250-256 A7 (Reserved for difference plotting)

GET\_DATA, TYPE = SOL\_PAR This command obtains solution tolerances, controls, or switches that influence the processing of the case in residence. The obtained system data is identical to the set of data modified by the related command /SOLUTION. The returned values are encoded in the character array in\_buffer in free field, C-formatted strings. The quantities enclosed in angle brackets "< ... >" denote variables quantified; < status > denotes a logical on or off; < value > denotes an integer, floating point, or character quantity. /GET\_DATA, TYPE = SOL\_PAR, > AI\_CONTROL = < value > { CON | MOD | OFF } > BASE\_SOLUTION = < status > > DEBUG\_TX = < status > > DEBUG\_BUS = < status > > DEBUG\_AI = < status > > DEBUG\_DC = < status > > LIMITS\_QRES = < value > > LIMITS\_PHA = < value > > LIMITS\_DA = < value > > LIMITS\_DV = < value > > LTC = < value > { ON | ON\_NV | ON\_NPS | OFF | ON\_DCONLY } > MISC\_XBUS = < value > { BPA | VMAX | WSCC } > MISC\_DCLP = < status > > MISC\_VFLAT = < status > > MISC\_TSTART = < value > > MISC\_ITER\_SUM = < status > > MISC\_PHA\_SHIFT\_BIAS = < value > { BPA | WSCC } > SOL\_ITER\_DECOUP = < value > > SOL\_ITER\_NEWTON = < value > > TOL\_BUSV = < value > > TOL\_AIPOWER = < value > > TOL\_TX = < value > > TOL\_Q = < value > return status: status = 0 : success 1 : errors

GET\_DATA, TYPE = SUB\_DEFINE This command performs character string substitution using computed base case quantities upon the tokens defined with the >DEFINE statement within comment records in the USER\_ANALYSIS command. It calls p\_subdef.f with the following parameters. integer function p\_subdef (in\_buffer, out\_buffer) parameter (MAXBUFFER = 6600) character in\_buffer \* (MAXBUFFER) character out\_buffer \* (MAXBUFFER) The character array in\_buffer contains the following information. /GET\_DATA, TYPE = SUB\_DEFINE, SOURCE = BASE

#### ALTERNATE\_BASE

GET\_DATA, TYPE = STATUS This command retrieves the case description. It calls gtstatus.f with the following parameters. integer function gtstatus out\_buffer) parameter (MAXBUFFER = 6600) character in\_buffer \* (MAXBUFFER) character out\_buffer \* (MAXBUFFER) The character array in\_buffer contains the following information. /GET\_DATA, TYPE = STATUS The character array out\_buffer contains the following information. c c Program: <n> version <n> date <n> c Program size: max buses <n> max branches c Case: <name> status <n> base file <n> c <n> buses <n> branches <n> areas <n> d-c lines c <n> changes c c comments c c comments c c comments c

GET\_DATA, TYPE = SYSTEM This command obtains system-specific information pertaining to parameters which describe general characteristics of the base case in residence. The returned values are encoded in the character array out\_buffer in free field, C-formatted strings. The quantities enclosed in angle brackets "< ... >" denote variables returned; < status > denotes a logical on or off and < value > denotes an integer or floating point quantity. /GET\_DATA, TYPE = SYSTEM, CASE\_DT = < case date > OLD\_BASE = < file name > NEW\_BASE = < file name > OLD\_NETD = < file name > NEW\_NETD = < file name > OLD\_CHGF = < file name > NEW\_CHGF = < file name > PRG\_VERS = < program version > BASE\_MVA = < base MVA > NUM\_DC\_SYS = < number of DC systems > NUM\_AREA = < number of areas > NUM\_ITIE = < number of interties > NUM\_ZONE = < number of zones > NUM\_OWN = < number of owners > NUM\_BUS = < number of buses > NUM\_AREA\_SBUS = < number of area slack buses > NUM\_DC\_BUS = < number of DC buses > NUM\_AGC\_BUS = < number of AGC buses > NUM\_BX\_BUS = < number of BX buses > NUM\_ADJ\_BUS = < number of adjustable buses > NUM\_PCT\_VAR\_BUS = < number of % VAR controlled buses > NUM\_BRN = < number of branch records > NUM\_CKT = < number of circuits > NUM\_DC\_LINE = < number of DC lines > NUM\_LTC = < number of LTC transformers > NUM\_PHAS = < number of phase shifters > SOLN\_STATUS = < solution status > NUM\_KV = < number of different KVs > NUM\_REC\_TYP = < number of record types > return status: status = 0 : success 1 : errors

GET\_DATA, TYPE = ZONE\_LIST This command loads the zone list dialog in the Reports Dialog and the Network Data Edit Dialog. See the IPF Basic User's Guide. It returns in out\_buffer the list of filtered zones in the following format. <zone>LINEFEED

<zone> is the zone name in A2 format. It calls zone\_list.f with the following parameters. integer function zone\_list (in\_buffer, out\_buffer) parameter (MAXBUFFER = 6600) character in\_buffer \* (MAXBUFFER) character out\_buffer \* (MAXBUFFER) The character array in\_buffer contains optional filter data in the following command. /GET\_DATA, TYPE = ZONE\_LIST [ FROM BUS\_DATA ]

WHERE AREAS = <area1>, <area2>, etc AND ZONES = <zone1>, <zone2>, etc AND OWNERS = <own1>, <own2>, etc AND BASEKV = base1 < base ( example < 115.0 means all base kv's less than or equal to 115.0) > base ( example > 115.0 means all base kv's greater than or equal to 115.0) base1 < base2 (all bases between base1 and base 2) base2 > base1 (same as above) TYPE = '\*' , 'A\*', 'A?', 'I' , 'B\*', 'L\*', 'B?',



'B ', 'BE', 'BS', 'BC', 'BD', 'BV', 'BQ',  
 'BG', 'BT', 'BX', 'BM', 'BF', '+ ', 'X ', 'Q ', 'LD', 'LM', 'E ', 'T ', 'TP', 'R ', 'RZ' BUS = "<busname>" (quotes  
 are necessary) AFTER\_BUS = "<busname>" ALL LOADING = (<min> <max>)

Details of the filter are found in the IPF Basic User's Guide under the section on "Dynamic Filters."

## 2.7.5 PUT\_DATA

PUT\_DATA, TYPE = COMMENTS This command replaces case comments, along with caseid, case description, and headers. There is a related command

/GET\_DATA, TYPE = COMMENTS

which obtains the corresponding data including header 1 which is not modifiable. Header 1 is formatted to include case name, case description, program version, date, etc. Up to 20 comments are allowed. The two header records must be present even if blank. For all blank "H" or "C" records (blank "C" records are accepted, but optional) include at least one blank character after the "H" or "C". The sent values are encoded in the character array out\_buffer in free field, C-formatted strings. The quantities enclosed in angle brackets "< ... >" denote variables. Headers may be up to 130 characters; comments may be up to 120 characters, not including the "H" or "C" in column 1. /PUT\_DATA, TYPE = COMMENTS CASE\_ID = "< case name >" 10 chars CASE\_DS = "< case description >" 20 chars H< header 2 information > H< header 3 information > C< comment text > ... C< comment text >

## 2.7.6 Report Generation

The /REPORTS command with its many different forms fetches data from the Powerflow process for display in the Reports Dialog. It calls p\_report.f with the following parameters. integer function p\_report (in\_buffer, out\_buffer) parameter (MAXBUFFER = 6600) character in\_buffer \* (MAXBUFFER) character out\_buffer \* (MAXBUFFER) The character array in\_buffer contains any of the following commands. /REPORTS, SELECT AI\_SUMMARY /REPORTS, SELECT BUS\_INPUT /REPORTS, SELECT BUS\_BR\_INPUT /REPORTS, SELECT BUS\_BR\_OUTPUT /REPORTS, SELECT BUS\_UVOV /REPORTS, SELECT LINE\_COMPARISON /REPORTS, SELECT NETWORK\_CHANGES /REPORTS, SELECT NETWORK\_DELETIONS /REPORTS, SELECT OVERLOADED\_LINES /REPORTS, SELECT OVERLOADED\_TXS /REPORTS, SELECT PHASE\_SHIFTER /REPORTS, SELECT TIE\_LINE\_SUMMARY /REPORTS, SELECT VOLTAGE\_COMPARISON The routine p\_report.f parses these command and calls a subroutine to perform the specific task. The modules are listed below.

### REPORTS, SELECT AI\_SUMMARY

This command retrieves filtered area interchange output data. It calls areaintrpt.f with the following parameters. integer function areaintrpt (in\_buffer, out\_buffer) parameter (MAXBUFFER = 6600) character in\_buffer \* (MAXBUFFER) character out\_buffer \* (MAXBUFFER) in\_buffer is positioned to the first character following the string AI\_SUMMARY. The character array in\_buffer contains the following information. /REPORTS, SELECT AI\_SUMMARY

[ OUTPUT = <filename> ]

The output is placed in out\_buffer. The report dialog is shown below.

AREA INTERCHANGE SUMMARY							
Area interchange summary case				date 6/11/93			
Area name	Slack bus	Slack bus generation					net
	Name	Base	Beginning	Change	Final	maximum	interchange
			MW	MW	MW	MW	MW
AREA 1	G31	13.8	1277.0	0.0	1277.0	1900.0	1105.0
AREA 2	G4	22.0	1641.0	0.0	1641.0	1200.0	-2210.0
AREA 3	G32	13.8	1277.0	0.0	1277.0	1900.0	1105.0

Close

Fig. 2.7.1: Area Interchange Summary Report Dialog

### REPORTS, SELECT BUS\_INPUT

This command retrieves filtered WSCC-formatted bus input data records. It calls businrpt.f with the following parameters. integer function businrpt(in\_buffer, out\_buffer) parameter (MAXBUFFER = 6600) character in\_buffer \* (MAXBUFFER) character out\_buffer \* (MAXBUFFER) in\_buffer is positioned to the first character following the string BUS\_INPUT. The character array in\_buffer contains the following information. /REPORTS, SELECT BUS\_INPUT [ FROM BUS\_DATA ]

[ OUTPUT = <filename> ] WHERE (repeat filter from BUS\_LIST)

The output is placed in out\_buffer. The report dialog is shown below.

BUS INPUT DATA											
Btc<0><<	Bus	><KV>ZN<Pl d><Ql d>Psh>Qsh><Pm>Pgen>qmax>qmin><vh><vl><rembus><kv>%q>									
BQ	G1	13.802				1200	1000	1000-1000	1010		
BX	G1X	13.802			300				10101010	G1X	13.8
BS	G2	13.802				1200	995.6		1000		
B	G2 HIGH	23002540.0	100.0								
BS	G31	13.801				1900	1277	1000-1000	940		
B	G31HIGH	50001100.0									
BQ	G32	13.803				1900	1277	1000-1000	940		
B	G32HIGH	50003100.0									
BG	G4	22.002				1200	1641	500.0 -500		G4 HIGH	230
B	G4 500	50002									

Close

Fig. 2.7.2: Bus Input Data Report Dialog



BUS BRANCH OUTPUT										
BUS NAME	VOLTS-PU	...	GENERATION..	...	LOAD.....	...	shunt.....	...	bus	
							avail	used	type	
ACTUAL KV/ANGLE			MW	MVAR	MW	MVAR	mw	mvar	mvar	
G2	13.8 1.000		988.6	0.0	0.0	0.0	0.0	0.0	0.0	BS
	13.8/ 0.0									
-----ID TO BUS NAME										
			LINE FLOWS			line losses		....loading		
			MW	MVAR		mw	mvar	pct	actual	
13.20/230.00		G2 HIGH 230	988.6	40.8		0.0	121.8	90 N	991 MVA	
BUS NAME	VOLTS-PU	...	GENERATION..	...	LOAD.....	...	shunt.....	...	bus	
							avail	used	type	
ACTUAL KV/ANGLE			MW	MVAR	MW	MVAR	mw	mvar	mvar	
G2 HIGH 230	1.048		0.0	0.0	540.0	100.0	0.0	0.0	0.0	B
	241.1/ -7.0									
-----ID TO BUS NAME										
			LINE FLOWS			line losses		....loading		
			MW	MVAR		mw	mvar	pct	actual	
230.00/ 13.20		G2	13.8	-988.6	81.0		0.0	121.8	90 N	991 MVA
Close										

Fig. 2.7.4: Bus Branch Output Report Dialog

**REPORTS, SELECT BUS\_UVOV**

This command retrieves filtered under/over voltage bus output data. It calls busuvovrpt.f with the following parameters. integer function busuvovrpt (in\_buffer, out\_buffer) parameter (MAXBUFFER = 6600) character in\_buffer \* (MAXBUFFER) character out\_buffer \* (MAXBUFFER) in\_buffer positioned to the first character following the string BUS\_UVOV. The character array in\_buffer contains the following information. /REPORTS, SELECT BUS\_UVOV [ FROM BUS\_DATA ]

[ OUTPUT = <filename> ] WHERE (repeat filter from BUS\_LIST)

The output is placed in out\_buffer. The report dialog is shown below.

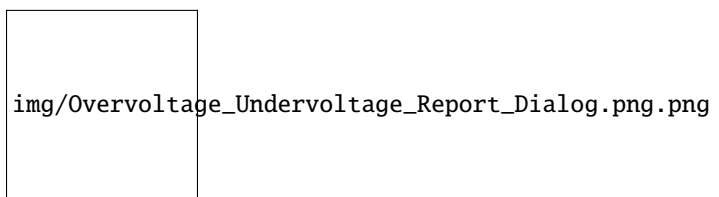


Fig. 2.7.5: Overvoltage Undervoltage Report Dialog

**REPORTS, SELECT LINE\_COMPARISON**

This command retrieves filtered line loading differences between the base case in residence and a selected base case history data file. It calls lfodifprt.f with the following parameters. (Not currently working.) integer function lfodifprt (in\_buffer, out\_buffer) parameter (MAXBUFFER = 6600) character in\_buffer \* (MAXBUFFER) character out\_buffer \* (MAXBUFFER) in\_buffer is positioned to the first character following the string LINE\_COMPARISON. The character array in\_buffer contains the following information. /REPORTS, SELECT LINE\_COMPARISON

[ OUTPUT = <filename> ] FILE = <filename> WHERE (repeat filter from BUS\_LIST)

The output is placed in out\_buffer.

**REPORTS, SELECT NETWORK\_CHANGES**

This command retrieves the list of all accumulated changes performed on the base case in residence. It calls `chglisrpt.f` with the following parameters. integer function `chglisrpt` (`in_buffer`, `out_buffer`) parameter (`MAXBUFFER` = 6600) character `in_buffer` \* (`MAXBUFFER`) character `out_buffer` \* (`MAXBUFFER`) `in_buffer` is positioned to the first character following the string `NETWORK_CHANGES`. The character array `in_buffer` contains the following information. `/REPORTS, SELECT NETWORK_CHANGES`

[ `OUTPUT` = <filename> ]

The output is placed in `out_buffer`. The report dialog is shown below.

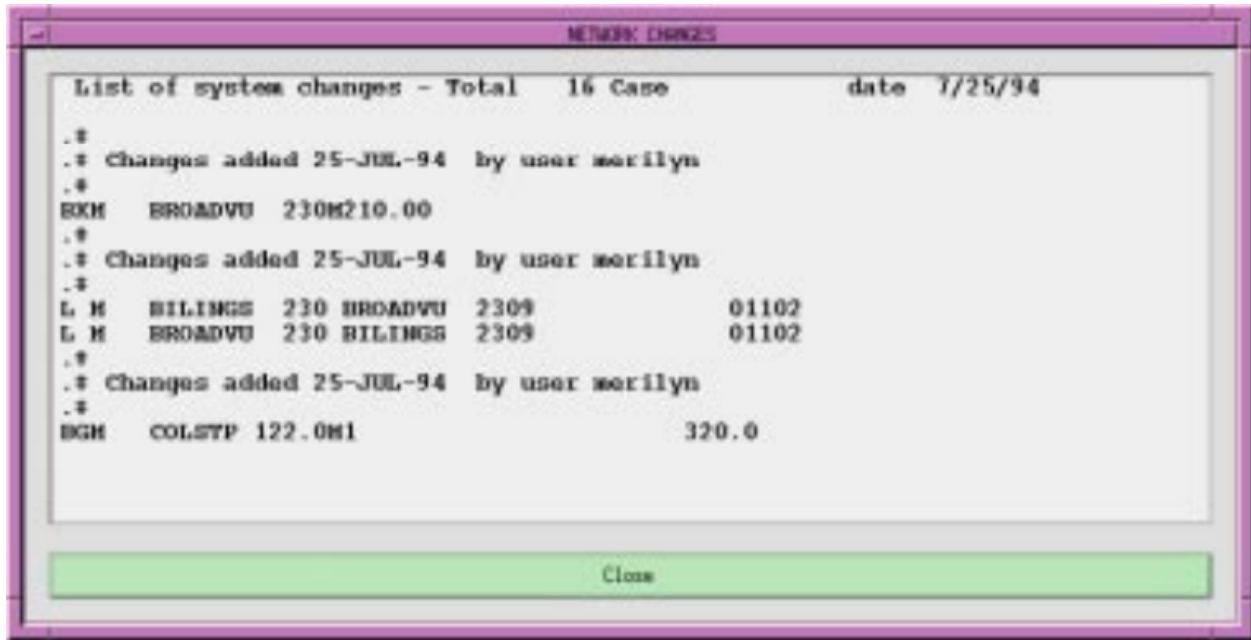


Fig. 2.7.6: Network Changes Remote Dialog

**REPORTS, SELECT NETWORK\_DELETIONS**

This command retrieves the list of all deleted network data in WSCC format. It calls `deleterpt.f` with the following parameters. integer function `deleterpt` (`in_buffer`, `out_buffer`) parameter (`MAXBUFFER` = 6600) character `in_buffer` \* (`MAXBUFFER`) character `out_buffer` \* (`MAXBUFFER`) `in_buffer` is positioned to the first character following the string `NETWORK_DELETIONS`. The character array `in_buffer` contains the following information. `/REPORTS, SELECT NETWORK_DELETIONS`

[ `OUTPUT` = <filename> ]

The output is placed in `out_buffer`. The report dialog is shown below. (This is the report accessed under Bone Pile.)

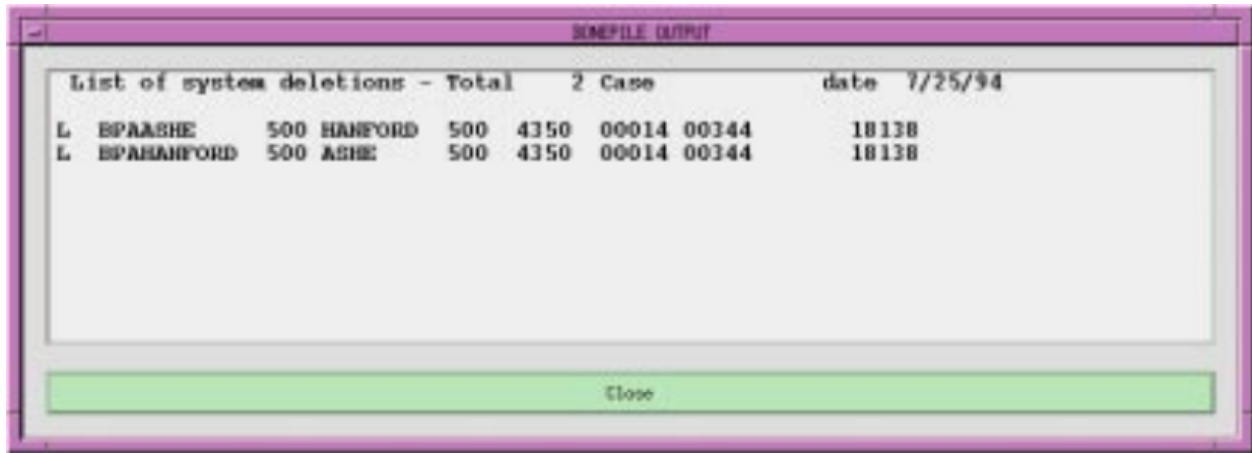


Fig. 2.7.7: Bonepile Output Report Dialog

### REPORTS, SELECT OVERLOADED\_LINES

This command retrieves filtered overloaded branch output data. It calls `ovldlnsrpt.f` with the following parameters. integer function `ovldlnsrpt` (`in_buffer`, `out_buffer`) parameter (`MAXBUFFER = 6600`) character `in_buffer` \* (`MAXBUFFER`) character `out_buffer` \* (`MAXBUFFER`) `in_buffer` positioned to the first character following the string `OVERLOADED_LINES`. The character array `in_buffer` contains the following information. `/REPORTS, SELECT OVERLOADED_LINES [ FROM BUS_DATA ]`

[ `OUTPUT = <filename>` ] WHERE (repeat filter from `BUS_LIST`)

The output is placed in `out_buffer`. The report dialog is shown below.

### REPORTS, SELECT OVERLOADED\_TXS

This command retrieves filtered overloaded transformer output data. It calls `ovldtxsrpt.f` with the following parameters. integer function `ovldtxsrpt` (`in_buffer`, `out_buffer`) parameter (`MAXBUFFER = 6600`) character `in_buffer` \* (`MAXBUFFER`) character `out_buffer` \* (`MAXBUFFER`) `in_buffer` is positioned to the first character following the string `OVERLOADED_TXS`. The character array `in_buffer` contains the following information. `/REPORTS, SELECT OVERLOADED_TXS [ FROM BUS_DATA ]`

[ `OUTPUT = <filename>` ] WHERE (repeat filter from `BUS_LIST`)

The output is placed in `out_buffer`. The report dialog is shown below

### REPORTS, SELECT PHASE\_SHIFTER

This command retrieves the phase shifter report. It calls `phshftrpt.f` with the following parameters. integer function `phshftrpt` (`in_buffer`, `out_buffer`) parameter (`MAXBUFFER = 6600`) character `in_buffer` \* (`MAXBUFFER`) character `out_buffer` \* (`MAXBUFFER`) `in_buffer` is positioned to the first character following the string `PHASE_SHIFTER`. The character array `in_buffer` contains the following information. `/REPORTS, SELECT PHASE_SHIFTER`

[ `OUTPUT = <filename>` ]

The output is placed in `out_buffer`. The report dialog is shown below.

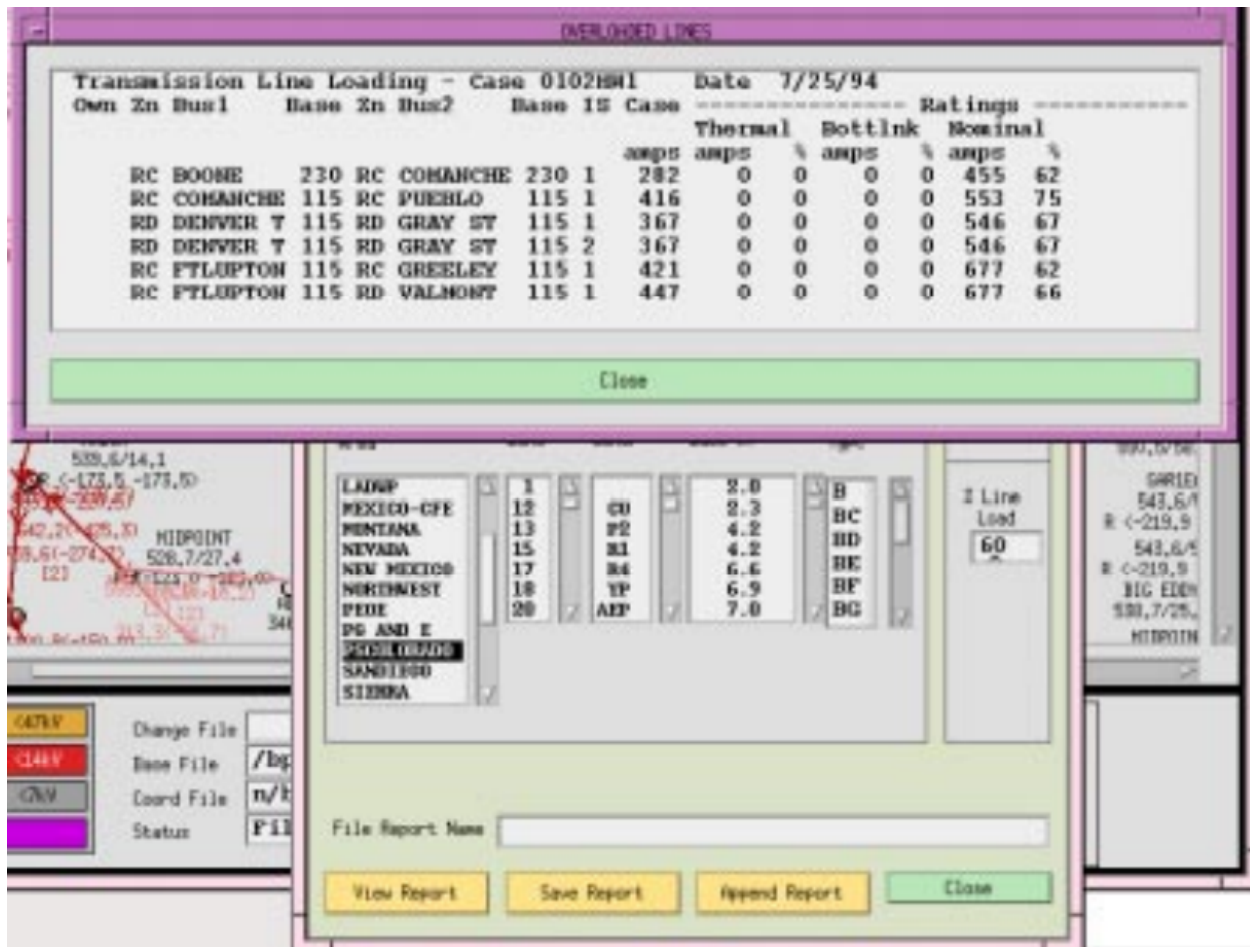


Fig. 2.7.8: Overvoltage Lines Report Dialog



Summary of phase shifters - Total 25 Case										date 3/25/94	
0-----	Phase-shifter		-Phase (degrees)-			Flow (MM)		dP/dT			
	P	S	tap	min	max	actual	min	max	(MM/d)		
BILINGS 161.0	HLGS PHA	161.0	.0	-60.0	.0	5.1	-36.0	-34.0	-.879		
BILINGS 230.0	HLGS PHA	230.0	.0	-60.0	.0	44.9	-70.0	-68.0	-.329		
CAL S PS 120.0	CAL SUB	120.0	-.0			-49.3					
CROS PHA 230.0	CROSVRUM	230.0	.0	-.0	75.0	-93.2	-39.0	-35.0	-.669		
DELTAPS 345.0	INTERSTA	345.0	-.0			-389.3					
FT CH PS 120.0	FT CHSR	120.0	-21.1			49.1					
GLEN PS 230.0	GLENCAMY	230.0	.0			142.2					
H ALLEN 345.0	HA PS	345.0	3.4	-60.0	60.0	102.1	95.0	105.0	-.455		
INVO 115.0	INVO PS	115.0	7.8	-30.0	30.0	50.0	50.0	50.0	-1.823		
JEFFERSH 161.0	JFRSHPHA	161.0	-.0			-34.6					
LADD 230.0	LACRANDE	230.0	.0			25.6					
LIBERTY 230.0	LIDTYPHS	230.0	-.0			-13.6					
MLY230 230.0	MLYPHS	230.0	0.0	-40.3	40.3	126.0	-400.0	400.0	-.240		

Close

Fig. 2.7.10: Phase Shifter Report Dialog



**REPORTS, SELECT TIE\_LINE\_SUMMARY**

This command retrieves filtered area tie line flows. It calls `inttierpt.f` with the following parameters. integer function `inttierpt` (`in_buffer`, `out_buffer`) parameter (`MAXBUFFER` = 6600) character `in_buffer` \* (`MAXBUFFER`) character `out_buffer` \* (`MAXBUFFER`) `in_buffer` is positioned to the first character following the string `TIE_LINE_SUMMARY`. The character array `in_buffer` contains the following information. `/REPORTS, SELECT TIE_LINE_SUMMARY`

[ `OUTPUT` = <filename> ] `WHERE` (repeat filter from `BUS_LIST`)

The output is placed in `out_buffer`. The report dialog is shown below.

Tie line summary of area interchange case								date
Area 1	Area 2	/----- Intertie branch -----/						6/11/93
		Zone	Bus 1		Zone	Bus 2	line flow	
OAREA 1	AREA 2	01	* G31HIGH	500.0	02	MALINA	230.0	7.9
		01	G31HIGH	500.0	02	* MID PS	500.0	1163.5
		01	RECT1AC	230.0	02	* G1	13.8	-500.0
		01	RECT1AC	230.0	02	* G1X	13.8	0.0
		01	RECT1AC	230.0	02	* G2 HIGH	230.0	-224.3

Fig. 2.7.11: Tie Line Summary Report Dialog

**REPORTS, SELECT VOLTAGE\_COMPARISON**

This command retrieves filtered voltage differences between the base case in residence and a selected base case history data file. It calls `vltdifprt.f` with the following parameters. (Not currently working.) integer function `vltdifprt` (`in_buffer`, `out_buffer`) parameter (`MAXBUFFER` = 6600) character `in_buffer` \* (`MAXBUFFER`) character `out_buffer` \* (`MAXBUFFER`) `in_buffer` is positioned to the first character following the string `VOLTAGE_COMPARISON`. The character array `in_buffer` contains the following information. `/REPORTS, SELECT VOLTAGE_COMPARISON`

[ `OUTPUT` = <filename> ] `FILE` = <filename> `WHERE` (repeat filter from `BUS_LIST`)

The output is placed in `out_buffer`.

## 2.8 Command Line Tools

### 2.8.1 bpf

Command line interface that performs power flow. It executes using commands from a Power Flow Control (PFC) file. Example usage: `bpf bench.pfc`. The PFC commands (.pfc) used with `bpf` allow for complete power flow runs including defining the network model and commands to perform various operations such as evaluating outages. The *Record Formats* section describes the network model records available and the *Power Flow Control (PFC)* section describes the PFC syntax and commands available.

In order to process a case, `bpf` requires a program control file and a valid set of base case data, which may be a composite of `NETWORK_DATA` and `BRANCH_DATA` formatted ASCII files or an `OLD_BASE` file from a previous run of `bpf`, and a `CHANGE` file.

The PFC file either contains data used for the solution, or names files containing such data. The solution data is optionally saved on the file named in the `NEW_BASE` command.

### Types of Processing

Input files used vary with the type of IPF processing, so it is important that you have a good understanding of the purpose of each type of file. Different program functions use the files to perform specific processes. Some major processes are:

- Basic Processing (POWER FLOW)
- Merge Base /MERGE\_BASE
- Network Reduction /REDUCTION
- Outage Simulation /OUTAGE\_SIM

Sample PFC file setups for each of the following solution processes are given in ??.

### Creating a New Base Case

The figure below depicts the initial way an IPF case is processed and how the output is saved on a NEW\_BASE file, which may then become an OLD\_BASE file for subsequent change studies. The contents of the print file (PFO) are defined by the P\_INPUT, P\_OUTPUT, and P\_ANALYSIS commands. Likewise, the contents of the fiche file (PFF) are defined by the F\_INPUT, F\_OUTPUT, and F\_ANALYSIS commands.

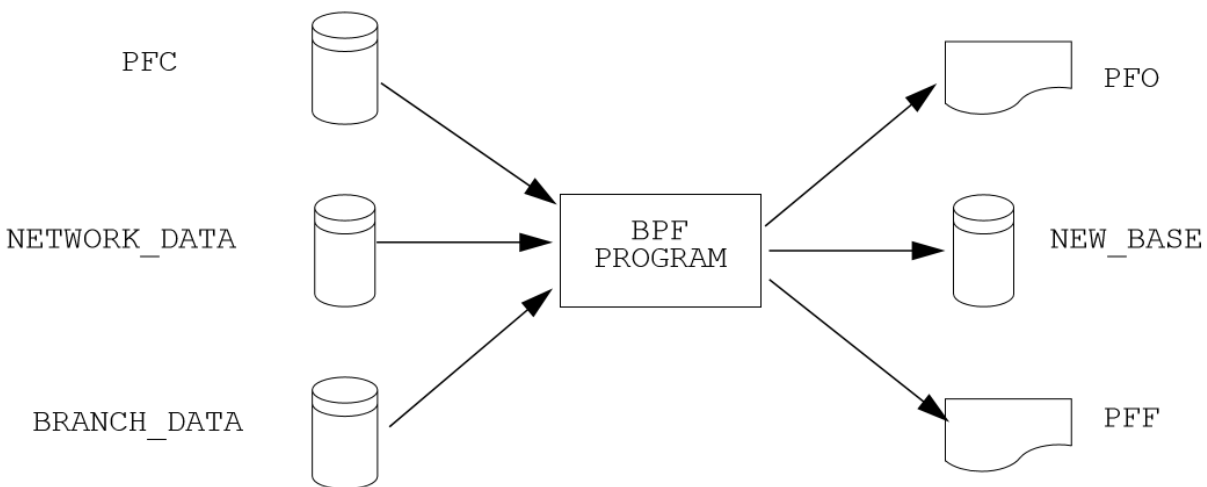


Fig. 2.8.1: New Base Creation Process

### Changing an Old Base Case

The figure below shows the most commonly used `bpf` process. A change case is created from an OLD\_BASE file using a `CHANGE` file. The modified case data is saved on the NEW\_BASE file. The output files PFO and PFF can be printed to paper or fiche or both.

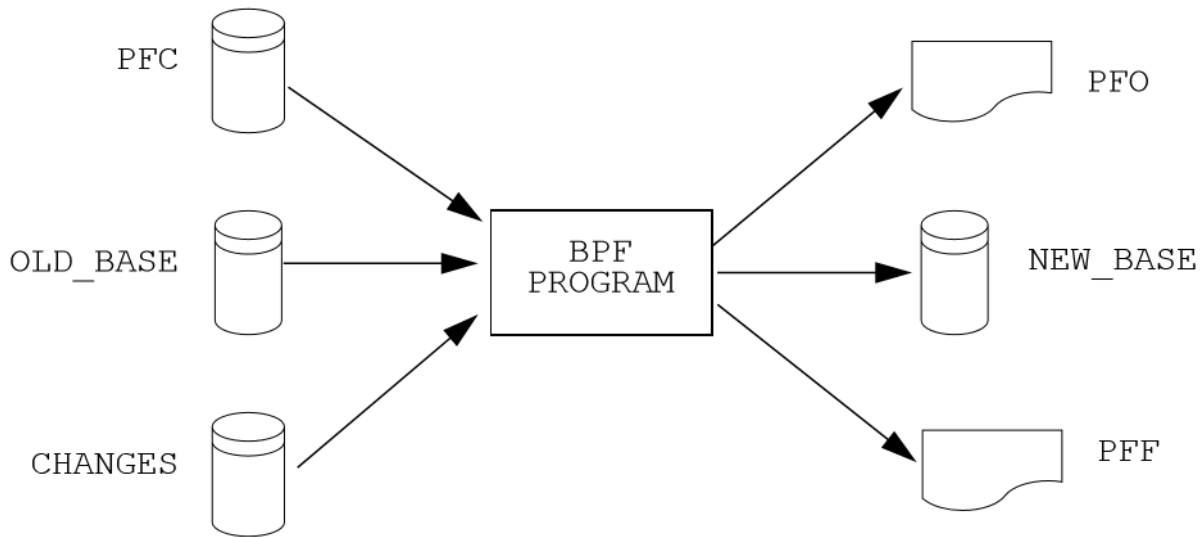


Fig. 2.8.2: Old Base Case With Changes

### Merging Subsystems

The figure below shows a `NEW_BASE` file being created by merging a subsystem from a case on an `OLD_BASE` file with another subsystem from either a second `OLD_BASE` or from a `BRANCH_DATA` and a `NETWORK_DATA` file. The output files `PFO` and `PFF` can be printed to paper and/or fiche.

### Reducing a Network

In the figure below, a network reduction is specified in the `PFC` file. Commands within this file define the retained system. The actual network reduction dynamically changes the base data in memory, and the reduced base case is saved on the `NEW_BASE` file. These output files (`.PFO` and `.PFF`) can be printed to paper and/or fiche.

For static reduction, you can use the `ipfcut` program. It is described in ??.

### Simulating Outages

The figure below shows an outage simulation being processed directly from an `OLD_BASE` file. Only printed analysis is output; no data files are generated. This printed output can be directed to either fiche or paper. Simulating outages is a special power flow function that subjects a subsystem of interest to a series of single contingency branch outages and tabulates the consequences of each outage or the cause of each overload.

## 2.8.2 ipf\_reports

`ipf_reports` is a command line tool that can be used to get information on an existing base case (`.bse`) file.

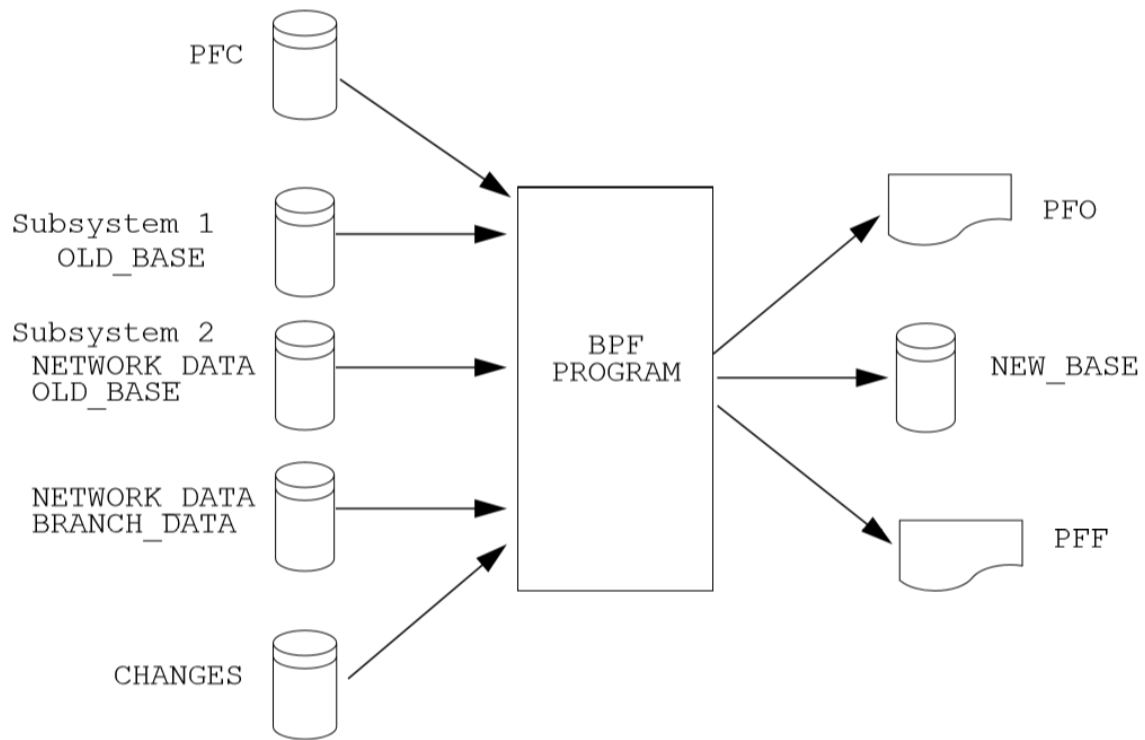


Fig. 2.8.3: Merging Two or More Subsystems

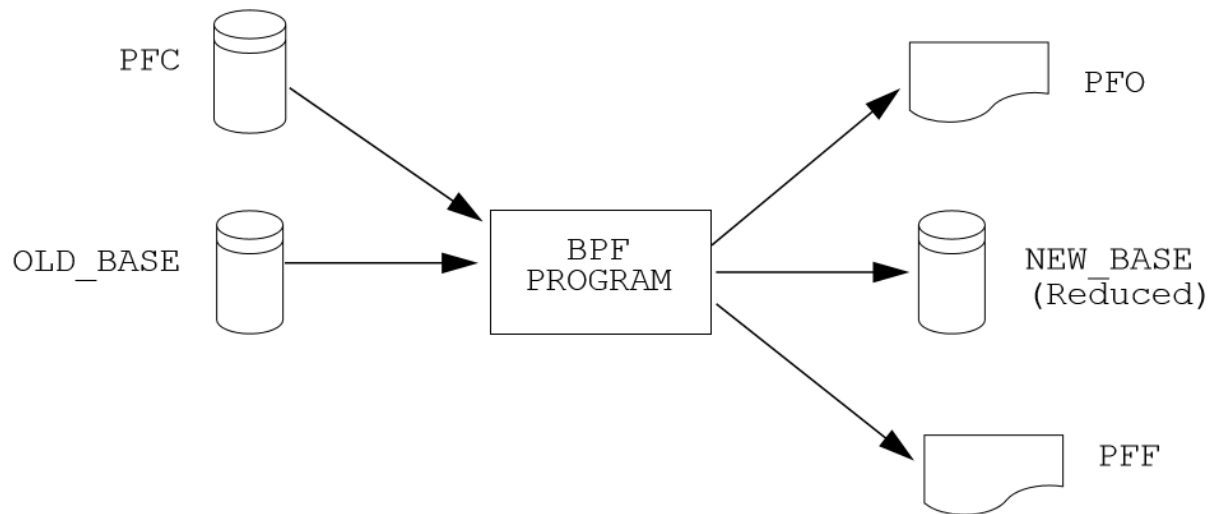


Fig. 2.8.4: Reducing a Network

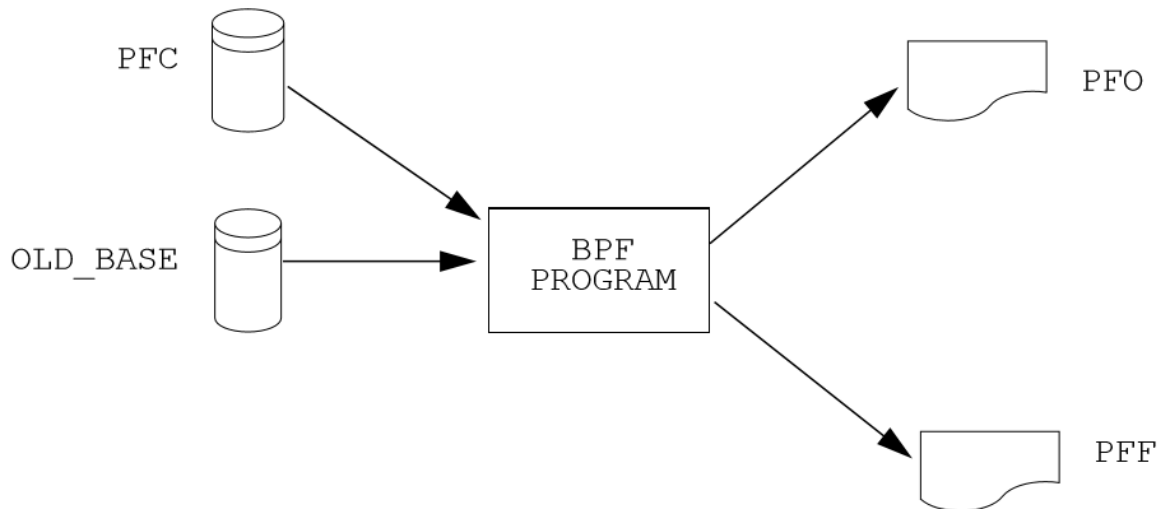


Fig. 2.8.5: An Outage Simulation

### 2.8.3 ipfplot

**ipfplot** is a batch plotting program to produce printed maps. The program accepts a coordinate file (`.cor`) and a base case file (`.bse`) on the command line, as well as an optional second base case file. When the second base case file is specified, a difference plot is produced. You can also use IPFPLOT to produce bubble diagrams. The same coordinate files are used for both GUI and IPFPLOT, but not all capabilities are available in the GUI.

### 2.8.4 netdat

**netdat** is a command line program that converts a binary base file (`.bse`) created by **bpf** to an ASCII network data file. It provides similar function to **ipfnet**, but **ipfnet** generates a network data file based on the currently loaded case in the GUI, rather than from a `.bse` file.

### 2.8.5 ipfcut

**ipfcut** is a command line program that cuts out a subsystem from a solved base case file. The full system resides in a base case file; the cut system is a card image Bus/Branch data file. Flows at the cut branches are converted into equivalent generation or load on specially formatted +A continuation bus records. An ensuing power flow run should solve with internal branch flows and bus voltages which are identical to those quantities in the original base case.

- Several methods are available to define the cut system: specifying individual buses, zones, base kVs, or individual branches.
- A pi-back feature replaces selected buses with a passive-node sequence (lines consisting of sections) with the original loads, generation, and shunts, pi-backed in proportion to the line admittances.

The function of **CUTTING** and **REDUCTION** are similar, but their methodologies are different. Both generate subsystems whose internal composition and characteristics are identical to that of the base case. **REDUCTION** generates equivalent branches, shunt admittances, and injections such that internal nodes still “see” the full system. **CUTTING** generates equivalent shunt admittances and injections such that internal nodes can determine that the boundary has changed and the external system has been cut out, even though the internal flows and nodal voltages are identical.

The CUTTING program mandates that the flow into the cut-out system is constant. This is valid for eliminating radial feeder circuits, but not for eliminating a strongly interconnected external network. In the latter case, REDUCTION yields a more responsive equivalent.

A simple criterion can be used to determine whether CUTTING or REDUCTION is more appropriate.

- Will a line outage or other major perturbation near the boundary of the retained subsystem and eliminated system significantly alter the flow between the two systems?

If the answer is *no*, the flow will not be significantly altered, then CUTTING is acceptable. (It is the author's opinion that REDUCTION is always superior.)

The CUTTING program is initiated by entering `ipfcut` at the keyboard after the computer displays the system prompt.

From this point on the operation is interactive. You should respond to the questions as they are asked.

### Cutting Methodologies

Two simple techniques are employed. Both may be used.

- **Cutting the eliminated branches.** In cutting, the active and reactive power flowing into a cut branch is replaced with an equivalent but fictitious load, which is appended to the terminal bus with continuation buses (+A).
- **Pi-backing loads of retained buses.** In pi-back, the loads and shunt susceptances on selected pi-back buses are distributed to neighboring terminal buses in proportion to their branch admittances. Only branch transfer susceptance is used (a good approximation when  $X \gg R$ ). Also, the pi-back bus may contain at most two branches. This corresponds with early reduction schemes. The quantities pi-backed are appended to the terminal buses on specially coded continuation cards (+A \*\*\*).

### Input Commands

The syntax of CUT commands conform to the convention that has been adopted for the other IPF programs that use commands.

```
[FICHE, copies=n]
(CUTTING, Project=name, case ID=name)
```

The qualifiers that select the subsystem and enable special options are listed below:

```
>DEBUG<
>EXCLUDE_BUSES<
>INCLUDE_BUSES<
>PI_BACK_BUSES<
>SAVE_BUSES...<
>CUT_BRANCHES<
>SAVE_ZONES...,SAVE_BUSES...<
>WSCC<
```

>WSCC< Enables the WSCC option. The default is no WSCC. Special processing is effected with this option:

1. Active power flowing from a cut branch into a bus is treated as a bus load under the WSCC option. Otherwise, it is treated as a load or generation depending upon the sign of the quantity.
2. Base kV fields omit the decimal point without the WSCC option. For example, a 115.0 KV appears as "115". Under the WSCC option, the same field appears as "115.". The WSCC Powerflow program interprets the base fields as character instead of decimal, and those two fields are unique! Lane 115. is in New Mexico but LANE 115 is in Oregon!

3. Line sections created from pi-back are consolidated into a single equivalent pi branch with the WSCC option enabled. Otherwise, the branch records are preserved (with the necessary name changes). There is one exception: If a step-up/step-down transformer-line transformer occurs, the branch is unconditionally made into an equivalent section.
4. Any branch in the cut list that has an INT in the ownership field has its flow transferred to a +A INT continuation bus instead of a +A \*\*\* bus.

>DEBUG< Opens the program debug file. Output appears on a file with extension .pfd. This is used only by the program developers.

>SAVE\_ZONES... ,SAVE\_BUSES...< Defines the retained network as all buses whose bases and zones both match the specified list. If SAVE\_BUSES is null or omitted, only zones are considered. Continuation cards begin with a + in column 1.

For example: >SAVE\_ZONES NA,NB,NC,ND,NE,NF,NG,NH, NI,NJ,NK,RM<

Any number of >SAVE\_ZONE...SAVE\_BASE< commands may be submitted. >SAVE\_BASES...< defines the retained network as all bases whose buses match the specified list. It is not necessary to type a decimal part unless it is part of the base kV, for example, 13.8 but not 3.46. Continuation cards begin with a + in column 1.

The system is initialized as an eliminated network. The following commands define the composition of the retained system. With the exception of

>CUT\_BRANCHES<, the effect of the commands may be repeated in any order. Their effects are overlaid.

>INCLUDE\_BUSES< >EXCLUDE\_BUSES< >SAVE\_BUSES< These commands introduce buses that are specified on bus records that follow (B in column 1). >SAVE\_BUSES is used to specify the entire cut system, bus by bus.

>INCLUDE\_BUSES is used to expand the cut system with individually named buses. This is used in context with >SAVE\_ZONES or >SAVE\_BUSES to provide more flexibility in the cut system.

>EXCLUDE\_BUSES is used to contract the cut system with individually named buses. This is used in context with >SAVE\_ZONES or >SAVE\_BUSES to provide more flexibility in the cut system.

A maximum of 1000 records are permitted. In the unlikely event that this is insufficient, the above command(s) may be simply repeated with an additional block of bus records.

>CUT\_BRANCHES< This command introduces branches that are specified on line records that follow (L, T, or E in column 1). A maximum of 500 cut branch records is permitted.

The CUT\_BRANCHES are oriented in the following order: retained bus, cut bus.

The cut system is defined in the following manner. Starting from the set of all cut branches, each bus on the cut side, which is in the eliminated system, is expanded one-adjacent by examining each branch connected to that bus. All branches that are not connected to any bus on the retained bus side are in the cut system. Those terminal buses are eliminated.

The first pass determines all buses 1-adjacent that are in the cut system. The process is repeated, starting with all buses 1-adjacent to the cut boundary to find all buses 2-adjacent. This process is repeated until no further expansion occurs in either system. The major advantage of this approach is that any incomplete cut enclosure is properly diagnosed near the missing branch.

If the WSCC qualifier is selected, any branch in the cut list that has an INT in the ownership field will have its flow transferred to a +A INT continuation bus instead of a +A \*\*\* bus. This is done so that if this cut system is to be reintegrated into another system the cut points can be easily identified and discarded.

Unlike other >...< commands, CUT\_BRANCH cannot be repeated.

>PI\_BACK\_BUSES< This process replaces a bus having one or two branches with an equivalent consisting of bus generation, load, and shunt admittances on the adjacent terminal buses.

If the bus originally had two branches, the new system has the following changes:

- The buses' generation, load, and shunt admittance are proportioned by the branch admittance to each terminal node.
- The bus is eliminated.
- The subsystem consisting of a bus and two branches is replaced with a single branch spanning the two terminal buses.

If the bus originally had one branch, the new system has the following changes:

- The buses' generation, load, and shunt admittance are transferred to the terminal node.
- The bus and its branch are eliminated.

In essence, a pi-backed bus becomes a passive node in a branch that now consists of sections. Since the quantities are pi-backed in proportion to their branch admittances, the redistribution approximates the effects of REDUCTION. A maximum of 1000 pi-back records may follow. If this limit is insufficient, the remaining pi-back records may follow another >PI\_BACK< command.

## Interactive Approach

The following is an example of the dialogue that occurs during an interactive execution.

```
* command file is: J8301FY84.CUT
ENTER NAME for BUS/BRANCH output file > J83CUT.DAT
ENTER file name for OLD_BASE > A8301FY84.BSE
```

### 2.8.6 pvcurve

pvcurve is a command line program that automates production of power (P) voltage (V) curve plot files and plot routine setup files for multiple base cases and outages.

### 2.8.7 post\_pvcurve

post\_pvcurve is a command line program that ?

### 2.8.8 qvcurve

qvcurve is a command line program that generates power-reactance curves.

### 2.8.9 findout

Command line interface that Generates a table of outages and corresponding branch overloads or bus voltage violations from power flow output (.pfo) files. Works with the .pfo output files of bpf runs that contain an / OUTAGE\_SIMULATION command. Runs as a post-processor to filter and sort the results and present them in tabular form. Tables of 'Outages and Overloads' or 'Outages and Bus Violations' can be produced. Entries in these tables can be filtered according to Zone, Owner, Base kV, Loading and Bus Voltage.

Tables can be sorted by Zone, Owner, Base kV, or alphabetically. The idea is to allow the user to automate the creation of a report detailing the results of outages instead of having to do manually which generally includes cut and paste operation with a text editor. Data fields in the output report table are character delimited to ease importing to Microsoft Excel or DECwrite.



### 2.8.10 lineflow

**lineflow** is a command line program that generates a table of values showing the requested branch quantities for multiple base cases. Selects lines by branch list, bus, kV, owner, zone, loading level, or matches to 'wild card' input. Sorts alphabetically, or by owner, zone, kV, loading (in percent), or according to input order of branches in a list. Generates a control script that allows repetitive similar studies to be performed automatically. Reports the following quantities: loading in Amps or MVA and percent of critical rating; or, power in, power out, and losses in MW. Data fields in the output report table are character delimited to ease importing to Microsoft Excel or DECwrite.

### 2.8.11 mimic

**mimic** is a command line program that generates new cases given a list of base cases and a list of change files. Check the new cases for over and under voltages, overloads, and excessive voltage and loading changes.

### 2.8.12 ipfsrv

**ipfsrv** is a service daemon which acts as the power flow server backend component of the *X Window Graphical Interface (gui)*. It executes *Powerflow Command Language (PCL)* commands dispatched from the gui. It gets launched automatically by the gui.

### 2.8.13 ipfbat

#### Overview

**ipfbat** is a command line program that is the batch version of **ipfsrv**. It accepts a Powerflow Control Language (.pcl) file. Plotting can be done with a control file; however, for most plots **ipfplot** is easier to use. Example of use: **ipfbat bench.pcl**. The PCL commands used with **ipfsrv** and **ipfbat** are described in *Powerflow Command Language (PCL)*.

#### Batch Mode Plotting

Batch mode plotting can be used when a coordinate file already exists, and the user simply wants a hard copy diagram based on that file and Powerflow data. If the Powerflow data is on a saved base case (\*.bse) file, the simplest method is to use the **ipfplot** program. However, **ipfbat** offers more flexibility and control. For example, with **ipfbat** you can load, solve, and plot a netdata file.

This technique can be used to produce diagrams that are generally produced through the GUI or for access to features that have not yet been implemented in the GUI. These features include plotting bubble diagrams, plotting difference diagrams, and plotting diagrams from a master list of coordinate files.

An example of batch mode plotting is accomplished through the **ipfbat** program as follows:

```
ipfbat bubble.pcl
```

where the .pcl file is a control file with the IPF commands and data necessary to produce a hard copy diagram.

Commands in the examples are record groups starting with a / (slash) command and ending with the next / (slash) command or (end) for the last command in the file.

Under the command /plot, the first line must name the coordinate file to be used, and the second must name the output PostScript file to be produced. Any subsequent records following, before the next /command, are interpreted as comments, and will be placed in the standard position following the last comment defined in the coordinate file.

Two special uses for comment records must be noted. If the record begins with an ampersand (&), it will be interpreted as an instruction to append the auxiliary coordinate file named on the record. At most one such file may be named. If the record begins with an 'at' symbol (@), it will be interpreted as an option record. Any diagram option indicated on this type of record will override the option specified in the coordinate file. Multiple @ records are allowed and will not be printed on the diagram.

### Example 1

Make a "standard" diagram (similar to the GUI operation):

```
/network_data,file=a92cy91.dat      ! Load the powerflow network data
/solution                          ! Solve the powerflow case
/plot                             ! Make a hard copy diagram
aberdeenmetric.cor                ! using this coordinate file
diagram.ps                        ! to build this postscript file.
Case prepared by: A. Perfect Planner ! Include this comment
Priority of study: RWI             ! and this comment
&aberdeeninset.cor               ! and this additional coordinate file.
@OPtion DIagram_type=Pq_flow      ! Supplement/Override *.cor options.
/syscal                           ! Hello operating system ...
lpr diagram.ps                    ! ... send this file to the printer.
/exit                             ! This job is finished.
(end)
```

### Example 2

Make a bubble diagram.

```
/old_base,file=j94cy91.bse ! Load the powerflow saved base case
/plot ! Make a hard copy diagram bubble.cor ! using this coordinate file
diagram.ps ! to build this postscript file. BUBBLE PLOT EXAMPLE
! Include this comment. /syscal ! Hello operating system ...
lpr diagram.ps ! ... send this file to the printer.
/exit ! This job is finished. (end)
```

### Example 3

Make a difference diagram:

```
/old_base,file=9_bus_test.bse      ! Load the powerflow saved base case
/get_data,type=load_ref_base,file=bus_alt1.bse ! Load a reference
                                         ! saved base case
/ get_data, type = load_ref_area    ! load reference solution data in tables
/plot                             ! Make a hard copy diagram
9bus_metricdif.cor                ! using this coordinate file
diagram.ps                        ! to build this postscript file.
Case prepared by: A. Perfect Planner ! Include this comment
Priority of study: RWI             ! and this comment
Difference plot between two cases   ! and this comment.
/syscal                           ! Hello operating system ...
lpr diagram.ps                    ! ... send this file to the printer.
/exit                             ! This job is finished.
(end)
```

### Example 4

Make a series of diagrams from a list of coordinate files.:

```

/old_base,file=/shr5/j96cy89.bse      ! Load the powerflow saved base case
/plot                                  ! Make a hard copy diagram
master.cor                             ! using all the coordinate files
                                      ! listed in this file
diagram.ps                             ! to build this postscript file.
Case prepared by: A. Perfect Planner   ! Include this comment
Priority of study: RWI                  ! and this comment on each diagram.
/syscal                               ! Hello operating system ...
lpr diagram.ps                         ! ... send this file to the printer.
/exit                                  ! This job is finished.
(end)

```

### Example 5

Here is an example of a master coordinate file (`master.cor`):

```

master
/home/dave/cor/3rdac.cor
/home/dave/cor/500bus.cor
/home/dave/cor/bubble.cor
/home/dave/cor/sworegon.cor
/home/dave/cor/nwmont.cor

```

## 2.8.14 ipf\_test

`ipf_test` is a command line program that provides an interactive way to run Powerflow Control Language commands. It is similar to `ipfbat` but prompts the user for input data rather than reading the power flow commands from a file.

## 2.8.15 ipfnet

`ipfnet` is the batch version of the “save netdata file” function built into the GUI / `ipfsrv`. This program generates a WSCC-formatted network data file in any of the following dialects: BPA, WSCC, or PTI. The GUI allows you to save a network data file describing the case you currently have loaded. This should not be confused with the `netdat` program, which performs very similar function by loading a saved base case (`.bse`) file and writing it out in an ASCII network (`.net`) file.

Both programs generate a WSCC-formatted network data file in any of the following dialects: BPA, WSCC1, or PTI. “Dialects” means that although the file is still WSCC format, the data is generated with special processing or restrictions and is destined for use with other programs. In the case of the PTI dialect, that data is intended to be processed by the PTI-proprietary conversion program `wscfor`.

This program extracts network data from a Powerflow “old base” history file. Table F-1 below summarizes the effects of each dialect.

Table 2.8.1: Effect of Dialects on Network Data File

Record or Field	Dialect	Effects
Header comments	PTI	<b>Three header records:</b> “<case_name>” “<case_name>” “<case_name> “
	BPA, WSCC, WSCC1	./CASE_ID = <case_name> ./CASE_DS = <case_description> ./H1 <header 1 text (auto-generated)> ./H2 <header 2 text (user input)> ./H3 <header 3 text (user input)> ./C001 <comment 1 text> ... ./Cnnn <comment nnn text>
Area “A” records	BPA, PTI	Encode zones 1-10 in “A” record, zones 11-20 in “A1” record, etc. Note: Voltage limits on “A” records are not encoded. They are specified by a default array that establishes limits using base kV and zones.
	WSCC, WSCC1	Encode only “A” record (any zones 11-50 will be lost). Note: Voltage limits on “A” records are not encoded. They are specified by a default array that establishes limits using base kV and zones.
Intertie “I” records	BPA, PTI	Single entry (low alpha to high alpha) associated “I” records follow each “A” record.
	WSCC, WSCC1	No “I” records encoded.
Default percentages on type BG buses	BPA	BG percentages are not changed.
	PTI, WSCC, WSCC1	BG percentages are calculated if their default value is invalid.
Continuation “+” bus records	BPA	“+” records are encoded.
	PTI, WSCC, WSCC1	“+” records are consolidated with “B” records.
Reactive capability “Q” records	BPA	“Q” records are encoded.
	PTI, WSCC, WSCC1	“Q” records are not encoded.
Minimum branch impedance	BPA, PTI	Branch impedances are not changed.
	WSCC, WSCC1	Minimum branch impedances are set to 0.0003 p.u.
Branch ratings	BPA	Options: 1. Use extended ratings (120-character records). 2. Replace nominal rating with minimum (Emergency, Thermal, or Bottleneck). 3. Use nominal rating only.
	PTI, WSCC, WSCC1	Options: 1. Replace nominal rating with minimum (Emergency, Thermal, or Bottleneck). 2. Use nominal rating only.
Branch sections	BPA	Encode as originally submitted.
	PTI, WSCC	Encode all branch sections in a consistent orientation.
216	WSCC1	Consolidate all sections into an equivalent branch
Regulating “R” records	BPA	Encode as originally submitted.
	PTI, WSCC	Options: 1. Encode as originally submitted.

The resultant output is an ASCII file. Two formats are available for the resulting output. The BPA format retains all of the extra features that are available in the BPA Powerflow program without making any modifications to the data, while the WSCC format option consolidates and restricts the features in order to be used with WSCC's IPS Powerflow program.

The CASEID of the power flow case data being extracted is used to create a file named CASEID.DAT. Any changes made to the data for WSCC (IPS) compatibility will be flagged on file CASEID.MES.

## Input

The `ipfnet` program prompts with the following requests:

- File name of the Powerflow "old base" history filename.
- Select output format desired: BPA, BPA1, BPA2, WSCC (IPS), or WSCC1 (IPS1).

## Sample Run

Type `ipfnet` at the system prompt and press the <RETURN> key. Answer the questions appropriately. An example is given below.

```
$ ipfnet
> Enter OLD_BASE file name (or Q to quit): ../dat/43bus.bse
> Enter name of network file (default is "../dat/43bus.net"): new.net
> Enter dialect (BPA, WSCC, WSCC1 or PTI): WSCC
> Enter record size (80 or 120): 80
> Nominal rating replacement code
  T = Thermal  E = Emergency  B = Bottleneck
  T: Transformers = T, Lines = T
  E: Transformers = E, Lines = T
  B: Transformers = B, Lines = B
  ET: Transformers = E, Lines = T
  EB: Transformers = E, Lines = B
  M: Transformers = min(TEB), Lines = min(TB)
> Enter rating replacement code: T
* Options selected - dialect = WSCC
                    size = 80
                    rating = T
> Are above options correct (Y or N)? Y
```

**Note:** The codes for dialect and rating must be upper case. `ipfnet` formats commands which are sent to `ipfsrv`. Some input checking is done, but invalid values may cause unexpected results.

### 2.8.16 ips2ipf

The *Record Formats* used by IPF are defined in ASCII format and consists of area, bus, and branch records. This format is very similar to the format used by the Western Systems Coordinating

Council (WSCC) back in the 1990s in their similarly named Interactive Powerflow System (IPS) application. However, note that IPF supports many record types which are not recognized by IPS, and in some cases the interpretation and application of the data values entered is different.

The `ips2ipf` command line program is designed to ease the burden of converting an IPS data deck into one which can be input to the IPF program with the expectation of getting the same powerflow solution results, within normal engineering tolerances. However, the conversion is not 100% automatic. See *IPS IPF Differences* section for more detail on the data input and internal modeling differences between the two programs.

Before running `ips2ipf` on an IPS data file, you should remove from the file all COPE commands (TITLE, NEW, ATTACH, etc.) The program will handle the standard 'control cards' HDG, BAS, and ZZ. Title records may be retained by putting an HDG in front of them, or by putting a period (.) in column 1 of each. An unlimited number of (.) comments are allowed, but these only annotate the data; they are not printed anywhere in the output.

`ips2ipf` performs the following tasks:

1. Renames duplicate buses.

IPS uses a 12-character bus name, which includes the base kV. IPF uses only 8 characters, plus the real value of the base kV. To IPS, `SAMPLE 230.` and `SAMPLE 230` are two different buses; to IPF they are the same bus.

`ips2ipf` identifies duplicate names and generates a different name for one of them. It reports any changed names; if you don't like the name it generated, you can change it after the fact.

2. Makes the system swing bus a BS bus, if given its name.
3. Transfers non-zero shunt vars from BE and BQ records to +A records.

In IPS, the shunt vars value entered for a bus which has variable var output is considered to be a fixed component of the total vars. In order to retain this philosophy in IPF, it is necessary to put the shunt on a +A (continuation bus) record. Shunt vars entered on the BE or BQ record are considered by IPF to be continuously variable.

4. Converts non-zero 'steps' on R records to 'taps' (by adding one).

IPS uses the number of steps available between TCUL taps; IPF uses the number of actual taps. If you run the conversion on an already converted file, another one will be added, which is probably not desirable.

5. Converts IPS comments (C in column 1) to IPF comments (. in column 1).

Unlike IPS, which prints the comments in the input data listing, IPF does not print them at all. But they can remain in the data file itself for information as long as they have a period in column 1 instead of a C.

6. Copies the controlled bus name from each X record to the corresponding BX record, to ensure that the proper bus is being controlled.
7. Copies the voltage limits from a BX record controlling a remote bus, to the remote bus record.
8. Corrects blank section id's in multi-section lines.

Blank is acceptable to IPF as a section identification only on single-section lines. `ips2ipf` identifies multi-section lines, and changes blank to 1, 1`` to ``2 etc. If there are actually 10 sections (IPS limit), then sections 8 and 9 will be combined and labeled 9.

9. Gives bus ties a small impedance.

IPF does not allow bus ties (0.0 impedance produces a fatal error.) `ips2ipf` changes this to  $(0.0 + j0.00001)$ , the same impedance IPF gives you when you sectionalize a bus and create a “bus tie” between the new bus and the old one. However, you should note that this may cause difficulties in getting a solution. (There are no zero impedance lines in standard WSCC study cases.)

10. IPF has no RF model. Any RF records in your deck will be ignored.

Items which are not handled by `ips2ipf`, which you need to look out for, are the following:

11. In IPS, line and bridge current ratings on DC are not processed, but only passed on to the Stability program. IPF actually uses them. You may find that the bridge current rating on the Intermountain DC line is too low.

## IPS IPF Differences

1. **Powerow Command Differences:** All IPF commands are different from those in IPS. When you are using the GUI, you will not have to worry about any of these, but there are some things you will need to do to your input data deck, such as deleting all the IPS control records and COPE commands (HDG, BAS, TITLE, ATTACH, \$DONE, END, etc.).
2. **Terminology:** The IPF Base Case (.bse) le is a binary le equivalent to the IPS History (.HIS) le. However, the Base Case le does not contain any mapping data, and only one case per le is permitted. The IPF Network (.net) le is an ASCII le equivalent to the IPS base case or base data le (.IPS). However, this le must not contain any modification records ('M' or 'D' in column 3). Changes go in a different le, which must be loaded separately. All mapping data is saved (in PostScript format) in a Coordinate le (.cor). Only buses which have a match in the currently loaded system data will be displayed.
3. **Case Title:** IPF builds the first line of a three-line IPS style title from the 10 character Caseid and the 20 character case description fields, and the other two lines from the two HEADER records. All of these are printed on standard BPA output listings, saved on the base case (history) file, and printed on hardcopy maps.
4. **Structure:** The IPF Changes file (.chg) contains new and modification records you want to apply in bulk to your base case (e.g. your own local system representation). You will use the GUI to make individual touch-up or particular study changes. The system slack bus must be specified as a 'BS' bus in the Network file; there is no GUI provision for selecting a slack bus (other than by changing the type of some bus to BS).
5. **Data Differences:** IPF system data is very similar to that for IPS, but is *not* identical. If you try to read in a WSCC base case deck as an IPF network file, you can expect numerous data errors and no solution. If you charge ahead, fixing fatal errors as you stumble over them, you will still probably not get the answers to match, because of modeling differences. The data conversion program handles most of these. There are two categories of differences between BPA and WSCC power flow models:
  - a. Modeling differences (including BPA extensions).
  - b. Input data differences

	WSCC's IPS	BPA's IPF
1	The DC line current rating is used only as a base by IPS. Both line current and bridge current ratings are passed to the Stability program; they are not used as limits in the powerflow solution.	The minimum of the bridge current rating and the line current rating is used as a limit by the DC system solution
2	Type RM phase shifters (controlling $P_{km}$ between $P_{min}$ and $P_{max}$ ) will bias the phase shift angle towards the original phase shift angle.	Type RM phase shifters (controlling $P_{km}$ between $P_{min}$ and $P_{max}$ ) will bias the phase shift angle to zero degrees to minimize circulating real power flow. WSCC bias is available as a solution option on the GUI.
3	A type BG generator may control only bus type BC.	A type BG generator may control bus types BC, B, BQ, BV, and BT.
2	An LTC may control only bus type BT.	An LTC may control bus types BC, B, BQ, BV, and BT.
5	Only one voltage control strategy per bus.	A generator and an LTC may simultaneously control a common bus. If a degree of freedom exists, the LTC will control $Q_{km}$ directly to minimize transformer var flow between terminal buses.
6	Type BX buses will bias the solution towards the original $X_{shunt}$ .	Type BX buses bias the solution to $V_{max}$ . WSCC bias is available as a solution option on the GUI.
7	Infinite default limits are assigned to type BG buses.	Default global voltage limits are assigned to all buses, including type BG buses, by base voltage level.
8	The bus shunt reactive on type BQ buses is fixed.	The bus shunt reactive on type BQ buses is continuously adjustable (0 to 100%). To make that quantity fixed, it must be entered on an accompanying +A continuation bus record. The conversion program automates this.
9	Inductance (G-jB) is applied to only one end of a transformer branch.	One half of G-jB is applied to each end of both transformers and balanced pi lines.
10	Model RF phase shifter takes several iterations to get from an initial angle to its final (fixed) phase shift angle.	No such model. Problems in solving phase shifters are handled internally.
11	Phase shifter must have same base kV at both terminals.	Step up phase shifter. Tap2 field is off-nominal tap2.
12	Phase shifter cannot be a section.	Phase shifting transformer can be a section.
13	Bus ties (zero impedance lines) receive special handling in solution and reporting.	No special bus tie model. A 'bus tie' is defined as a very low impedance line (0.0 + j0.00001).
14	Not available.	+ continuation bus records. Except for constant current load models, these records are used mainly for accounting purposes to differentiate generation, load, and shunt with unique owner-ships.
15	Not available.	I area intertie records. These records compute net area export on accompanying A records.
16	Not available.	A area record may be accompanied with A1, A2, A3, and A4 continuation records to accept a maximum of 50 zones per area.
17	Not available.	Branch records accept extended line current ratings: For types L and E records, thermal and bottleneck ratings. For types T and TP records, thermal, bottleneck and emergency ratings.
220		Types BM and LM multi-terminal DC data.
18	Not available.	Type RZ RANI devices.
19	Not available.	
20*	Base kV field interpreted as A4 for iden-	Base kV field interpreted as F4.0. SAMPLE 20.0



- The conversion program will handle this item.

## 2.9 Transient Stability Program (tsp)

The `tsp` program performs dynamic (e.g. generator rotor angle, governor output, power system stabilizer output, etc.) simulation of a power system when disturbed from its steady-state condition under various perturbations such as three phase faults, line to line faults, line to ground faults, loss of generation, etc.

### 2.9.1 Program Description

There are basically two methods of simulation used in this program. All differential equations are linear and solved by the trapezoidal rule of integration. The network equations are solved iteratively using the triangularized admittance matrix. Both of these methods are described in the paper *Fast Transient Stability Solutions* (IEEE Transactions on Power Apparatus and Systems, July/August 1972) written by Hermann Dommel and Nobuo Sato.

The diagram ?? shows that the swing program is divided into solution and output portions. The solution portion creates a swing solution file which saves all output data for all busses in the study. This structure allows the user to run the solution portion and save the swing solution file. Then, using the swing solution file, the power flow output file, and the swing output data file, the user can run any number of output jobs without resolving the solution.

The program is designed to run in conjunction with the IPF power flow program (`bpf`). The diagram ?? shows the interconnection between the power flow and transient stability program. Note that both the solution and output portions of the swing program require a power flow output file (`.bse`) as input.

The program also has a save data feature which allows the user to enter the majority of the swing input data via a save data file and enter only the line switching and FF records in the swing input file. This is useful when a series of swing studies are run using the same basic data but with different system perturbations.

Appendix C shows examples of how to create a save data file, how to run a study using a save data file, and how to run an output only job using the swing output file. The command files show how to make the necessary logical file assignments.

The DEBUG and COMMENT records are optional. Any information on the COMMENT records appears on the output listing as well as on the plot headings.

The switching records (LS) contain a variety of features and enable the user to properly define each event in chronological order. Several examples in preparing LS records are covered in ??.

The LZ record is used with the LS record and permits changing branch admittance quantities in the transient stability program.

The MZ record has the special feature which permits remote control signals for power system stabilizers and excitation systems. This card also inputs the generation transformer data for low-side generator bus determination not represented in the power flow but required for the excitation system.

Several control parameters and codes are available to the user and are entered in the DEBUG, CASE, FF, and 99 records. The CASE record and the 99 record have print options and the FF record contains several control parameters and codes for proper operation.

The output portion of the program provides the output data in the form of listings and/or plots. The plotting routines were written for a VERSATEC plotter and this code is not very portable.

If line impedances were modified during the solution portion of the program, to correctly output data for these lines the output portion of the program must be given the modified impedances via line modification (LM) records. ?? gives an example of how to use LM records.

The remaining information for the Transient Stability Program has not yet been migrated to ReadTheDocs format. See [TSP Users Guide PDF](#) for details.

## 2.10 X Window Graphical Interface (gui)

This section presents information about the X Window graphical user interface (gui) for IPF.

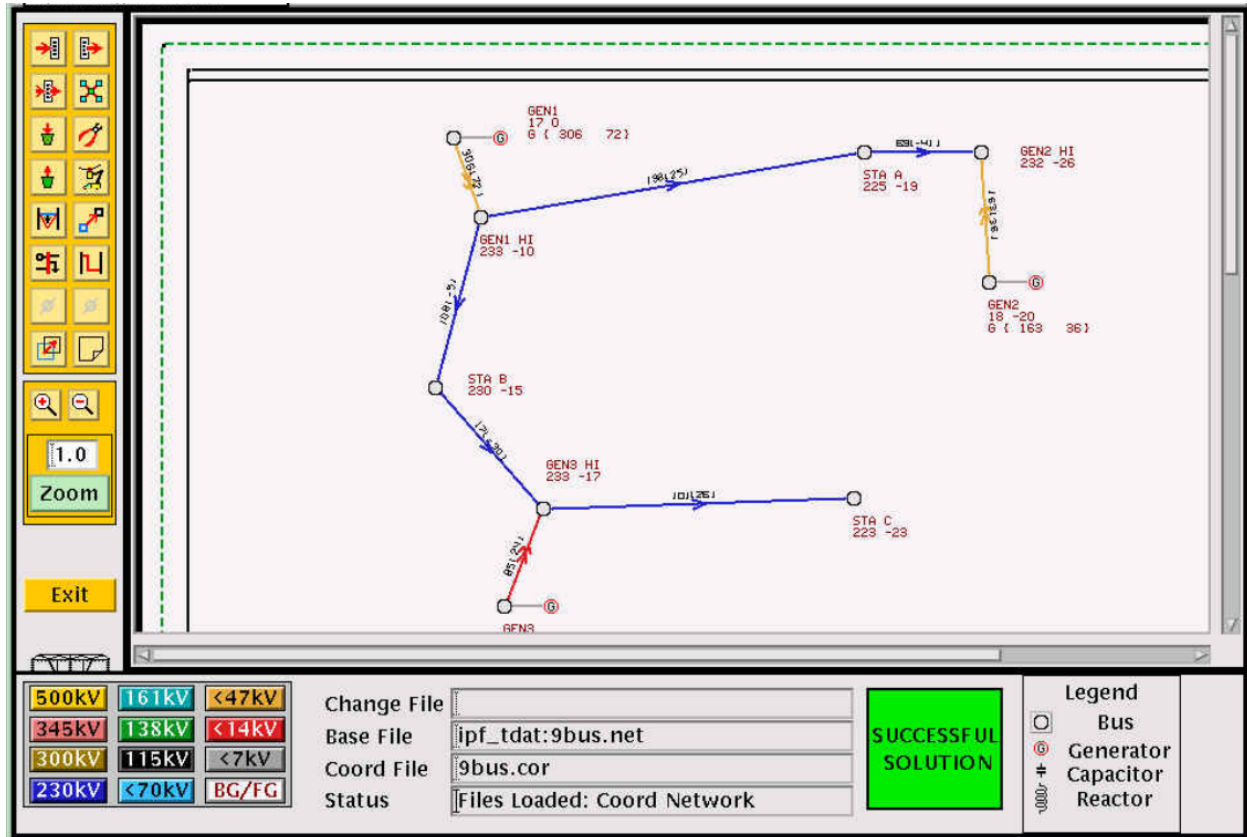


Fig. 2.10.1: Interactive Power Flow X Window GUI

IPF's X Window GUI interface makes data entry and modification easy. It also simplifies the running of base case solutions and the printing of network diagrams. This guide shows how to use the major features of IPF. Users who need details about data input record formats or system models should consult the [Record Formats](#) section.

IPF's GUI uses the X Window System and the OSF/Motif window manager interface. The graphical user interface features an interface that will be familiar to people who use an X Window System before, but it will feel a bit cumbersome compared to modern day applications.

## 2.10.1 Audience

You will be expected to already know the basics of power flow programs in general. You will find that small changes in bus and branch values are easier to make in the GUI than trying to manage through ASCII text files and running bpf commands on the terminal. For example, users will get a feel for how changes to the input data affect the solution voltages in a network much more quickly with the GUI than by using a terminal style interaction.

## 2.10.2 User Interface

The X Window System and the OSF/Motif window manager use certain descriptive terms for actions performed with the mouse. These terms are defined here. In addition, some keyboard actions are also defined. See the table below.

### Keyboard Conventions

The following conventions are used for key strokes. Generally, a hyphen (-) connects key names that should be pressed and held, starting from left to right. For example, the key sequence Control-c Shift-E means to press and hold the Control key and then press c. Release this combination and press and hold the Shift key and then the E key. Then release these keys. A different example: Esc f means to press and release the Esc key followed by the f key.

In general, the mouse operation and keyboard operation follow the conventions of the Motif interaction style guidelines. These are found in the OSF/Motif Style Guide. Many aftermarket books about Motif cover these conventions also.

### X Window System

This chapter provides a quick introduction to the X Window System and X window managers with emphasis on the OSF/Motif window manager. The treatment here is certainly not exhaustive or even complete. But it is intended to give you enough background to successfully use the Interactive Powerflow (IPF) program.

If you have not used an X Window based GUI before, be sure to go through this chapter for some pointers so that you'll be headed in the right direction. If you would like more information, refer to the following books resources the X Window System and OSF/Motif.

- Open Software Foundation. *OSF/Motif Style Guide Revision 1.1*. Prentice Hall, 1991. This provides the official description of OSF/Motif look, feel, and behavior for OSF/Motif software developers. Though not oriented toward OSF/Motif users, this book does give precise descriptions of all OSF/Motif components and behavior.
- Quercia, Valerie and Tim O'Reilly. *X Window System User's Guide OSF/Motif Edition*. O'Reilly & Associates, Inc., 1991. This is a good, general introduction to X and OSF/Motif.
- [Motif Programming Manual](#)

Broadly speaking, the X Window System is designed to deliver mouse-driven menu/window user interface applications over a local area network.

The X Window System specifies that the “look and feel” of its user interface be “policy free.” Because of this, programmers are free to create their own look and feel within broad limits. Over the past few years, Sun Microsystems, AT&T, and the Open Software Foundation have all created GUIs for the X Window System with a distinct look and feel. The Open Software Foundation offers OSF/Motif.

Like most large software systems, X and its environment have a jargon of their own. Here are a few terms you should know:

#### Server

The part of X resident in your local computer (or X terminal) memory. The server has three main tasks. First, the server takes care of communicating with the mouse and keyboard. Second, it takes care of managing X resources, such as fonts and colors. Third, it communicates with X applications.

### Client

A stand-alone X program. Clients are the X programs that you use to accomplish your work, such as drawing graphs, preparing text, making power flow calculations, etc. Clients usually reside in a computer across the local area network, but they can also reside in the same memory as the server itself. Clients and servers communicate through a special language (“protocol”) that is especially efficient for communication over a Local Area Network (LAN).

### Resources

X components that are held and managed in common for X clients by the server. X resources reside in your local computer (or X terminal) memory just like the server. X resources are things like fonts, color “pixmap,” font information, etc.

### Window manager

A special X client that handles the window services of other X clients. There are a number of different X window managers, not just one, but you only use one window manager at a time.

### X terminal

A special purpose computer having a built-in X server, and connection hardware and software for a LAN.

### LAN

A Local Area Network usually based on the hardware and low-level software standard of Ethernet. For Unix computers, the software communication protocols are usually based on the TCP/IP standard. PC networks may use different hardware and software network communication standards.

The GUI portion of IPF is a server; the powerflow portion is a client. These two programs may reside on different computers, in which case the GUI will be running on the machine you are sitting in front of, and the powerflow will be running on a workstation somewhere else, perhaps a network server. Note that the X Window terminology for “client” and “server” is reversed from the network communication terminology. It is also possible for both client and server applications to be running on the same computer.

The figure above shows you the basic architecture and communication model of the X Window System. Something you should note is that the client (application) program may be physically residing on a completely different computer from the one that your keyboard, mouse, and display are attached to. To access the remote application, you only have to know the name of the computer your client is on. (You also must have permission to use the other computer, of course, and it must be connected properly to the LAN.) The X Window System was designed from the ground up to run in a distributed computing environment.

When you are running a client, such as IPF, over the network, in contrast to running it in your own computer’s local memory, you will ordinarily notice very little performance degradation due to network traffic, though there may be some depending on how busy the network is. X is designed to minimize network communication.

The server side of X resides in your own computer’s (or X terminal’s) memory. There is one X server for each user’s keyboard, mouse, and display. The server is dedicated to you. However, your server may communicate simultaneously with many different X clients, not just IPF. And again, these other clients may be anywhere out on the LAN. Thus, in any one X session (between “login” and “logout”), you may run many X clients on many different computers. The server manages all this.

When your client requests a certain font, the server delivers it. When your client requests different colors for graphical objects, the server consults the color map for that client and delivers the correct colors. Fonts, colors, and certain other server-managed software components are termed resources. Some resources such as colors and fonts you can change in your own account’s IPF resources file, XGUI. See [Customizing the GUI \(XGUI\)](#) if you would like to learn more about this.

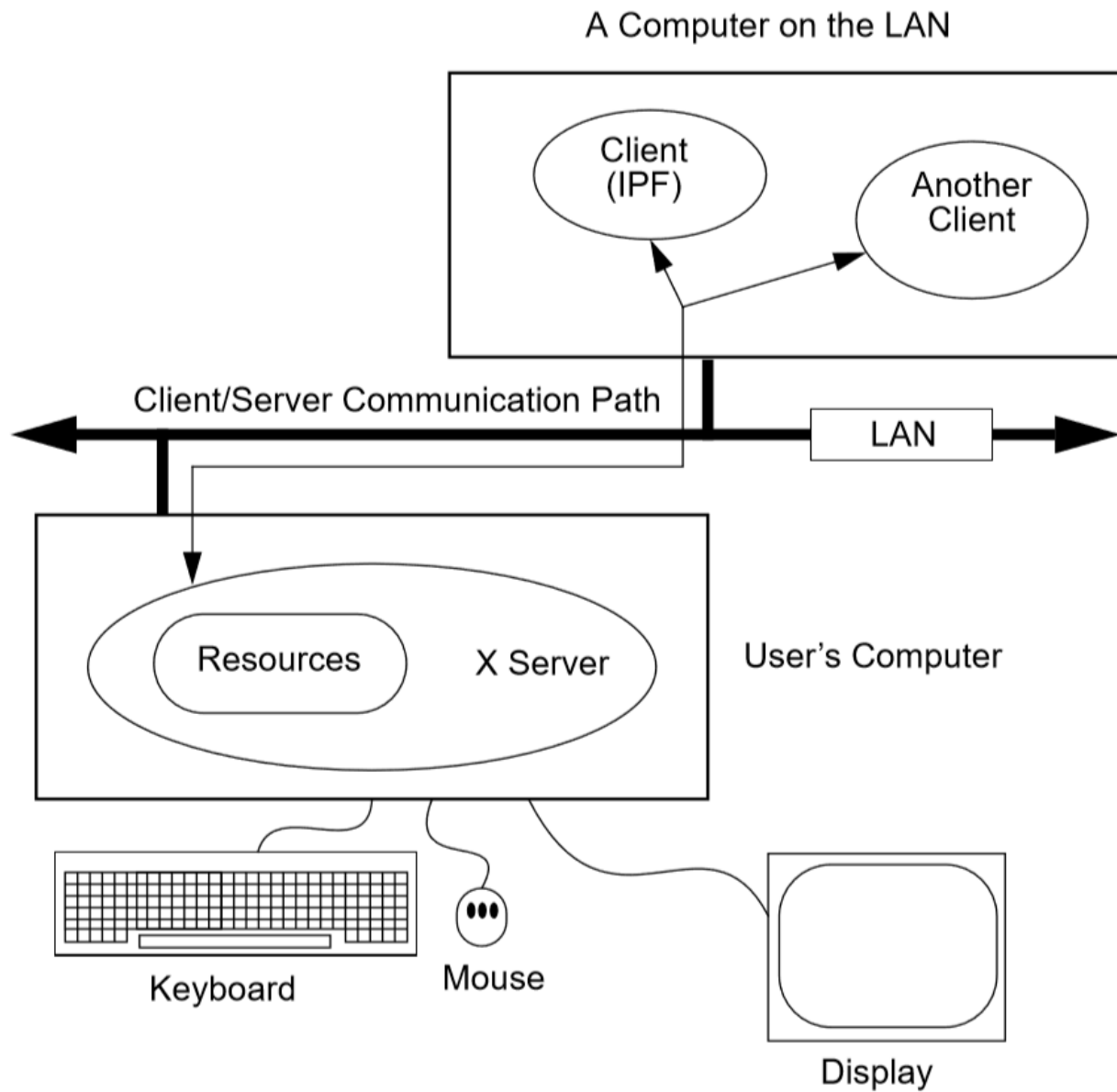


Fig. 2.10.2: A Server, Some Clients, and a LAN

### 2.10.3 A Summary of Motif Basics

IPF is designed to run and look best with the Motif window manager. This section introduces you to some Motif window manager basics. However, for detailed information, turn to the books cited at the beginning of this section.

#### Motif Windows

Motif windows are rectangular areas of the display. Various Motif components surround a central area where text and/or pictures appear. See example below. These components are controlled by the Motif window manager. However, the central area is controlled by an X client, which is a completely separate program from the Motif window manager. IPF gui is an X client, so it controls only the interior of Motif-managed windows. Because of this separation, you may notice that IPF's windows can still be moved, iconized, etc., even though the IPF client may not be responding.

In the figure below, the Terminal Emulator client controls the central display (where the text is), the Menu Bar, and the Scroll Bar. The Motif window manager (client) controls all the rest of the window.

#### Motif Resources

Like the X Window System, Motif has resources that you can change. Resources are system controlled components such as fonts, colors, initial size and position of windows, etc. Many X clients (application programs) have customizable resources. Since the Motif window manager is just another client, it has customizable resources, too.

Resources can be changed in two ways:

- Change the dedicated client resource file with an ASCII text editor. On Unix systems, the Motif resource file is named `.mwmrc` and the X resources file is `.Xdefaults`
- Use a dedicated X application that provides a regular GUI interface for changing the resource file.

Since interpreting the meaning of the resource specifications is not always easy or straightforward, it is recommended that you look for a dedicated X client for changing resources. Your Motif system probably has this X client already available, so that all you have to do is choose the Motif resource editor as a command on a system menu. Possible names to look for are System Setup, Configuration, User Preferences, or something similar. The editing of the many resources may be broken out on your system menus as separate commands, such as Colors, Fonts, Sizes, etc.

As a second choice, use a Motif book to help you interpret the meaning of the resource specifiers in the `.mwmrc` file and edit them with an ASCII editor such as the vi ASCII editor. Nearly all Unix systems have vi.

#### Common Windows Tasks

This section goes through a few common window and menu tasks to give you a feel for the Motif interface. Some tasks can be done through a menu command and through direct manipulation of a graphic component. And, in addition, many tasks can be accomplished through a sequence of keys without recourse to mouse movement or button clicks. The following brief descriptions concentrate on direct graphic component manipulation since this is usually the quickest way to get something done in Motif.

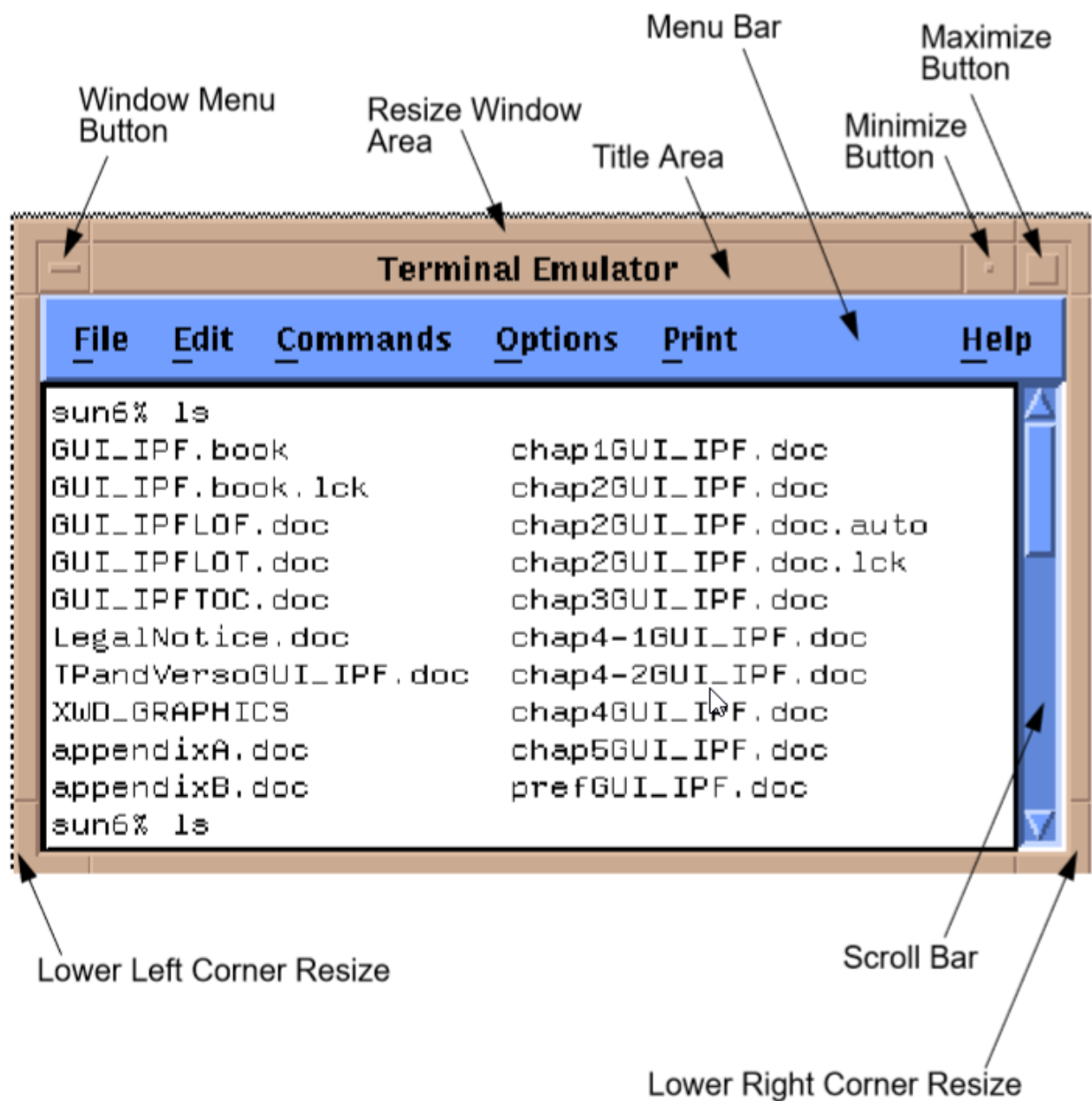


Fig. 2.10.3: Motif Window Components

### To move a window

Motif windows have a title area at the top. The window shown above has “Terminal Emulator” in the title area. You move Motif windows by “grasping” the window in this area and dragging it to a new location.

1. Move the mouse cursor to any point within the title area.
2. Press and hold the left mouse button.
3. Move the window to another location of the display. You will note that an outline box of the window shows you the dimensions of the window as you are moving.
4. Release the button. The window appears at the new location.

### To reduce a window to an icon

Icons are small, rectangular graphic objects that represent the main windows of Motif applications. Icons have many, but not all of the same attributes as windows - for example, they can be moved like windows. Icons are used to organize the display and reduce clutter. You are free to put icons wherever you like on the display. Some Motif systems organize icons in a special window, which looks like a desktop.

1. Click the Minimize button at the top right of a window. This shrinks the window to an icon and automatically places the icon at a predetermined place on the display. (Note that this place may be obscured by other windows!)
2. Move the icon as you would a window by placing the cursor over the icon, pressing the left mouse button, moving, and releasing the button.

### To change an icon into its window

An icon can be changed back into its windows by double clicking on it. The window will “remember” where its previous position was. Sometimes the timing between clicks is important, so make sure you do it fast enough.

1. Find an icon (or create one) and move the mouse cursor over the icon. Double-click the left mouse button. Be sure to keep the mouse cursor stationary between clicks; otherwise, Motif may interpret your actions as a “move icon” operation.
2. If you find that a menu pops up, select the Restore command by moving the cursor over the word and clicking once.

### To resize a window

Motif windows have a narrow border that acts as a “handle” for resizing operations. There are eight parts to the border — four corners and four sides. The corners are used to resize simultaneously both adjacent sides, and the sides are used to resize just one side at a time. The following procedure describes a common resizing operation.

1. Move the mouse cursor over the lower right corner. Note that this corner is demarcated by two cross cuts on the border a short distance away from the corner. The cursor may change its appearance when it is in this area, indicating that it is in the right position.
2. Press and hold the left mouse button on the lower right corner.
3. Move the mouse to the inside or outside of the current window.
4. Release the mouse button when you are satisfied with the size.



The opposite (upper left corner) remains stationary while you move the mouse around. A border line for two sides shows up to give you an idea of the size of the window as you move. Also, a pixel counter of the vertical and horizontal dimensions shows up to give you feedback if you need it.

If the corner or side the you want to grab is not visible on the screen, move the window until it is.

### To enlarge a window quickly

The Maximize button in the upper right corner is a quick way to enlarge a window to the maximum size of your display.

1. Move the mouse cursor over the Maximize button in the upper right corner of a window.
2. Click the button. Note that the window now covers the maximum area of the display. (All other windows should be covered.) You can resize the window using the directions above.

### To pop up a window's menu

All Motif windows have a minimum set of window management functions available in the upper left corner via the Window Menu button. Many of the functions described above are available through this menu. Some additional ones are there too, such as Close, Restore, and Lower.

1. Move the mouse cursor over the Window Menu button in the upper left corner.
2. Press and hold the left mouse button. Note that a menu pops up. Select one of the commands on the menu by moving the mouse cursor over the command and releasing. Only commands that are clearly visible are selectable. Those which are “grayed out” are not accessible or appropriate in the current context.

Closing a window causes the window to disappear from the display, so be careful because it can also quit the application. In the beginning, you may not know how to restart an application! (Look for a Motif window called the Session Manager. Your system operator has probably set up your account with a Session Manager. See if you can find the closed application on a menu in your Session Manager, so that you can re-open the application if you accidentally close it.)

Lowering a window means to cause it to go to the bottom of the windows “stack.” Think of Motif windows as pieces of rectangular paper on a desk. The ones that overlap have an order from top to bottom. The Lower command causes movement of windows in the stack from top to bottom. Experiment with the command to see how it works. You bring “buried” windows to the top by pushing visible ones down with the Lower command. Note, however, that if you see any piece of a Motif window, you can just click on or inside the border to bring it immediately to the top.

## 2.10.4 IPF as an X Client

IPF is built on top of the X Window System and uses the facilities of an X window manager of your choice. (However, the recommended window manager is Motif.) In the figure below, you can see that both the X server and the GUI part of IPF reside on the user's computer. Another client that is always present on the user's computer is the user's window manager. This is not shown in the figure below, however. In most configurations, the “engine” (`ipfsrv`) part of IPF resides on the user's computer, but may, as shown below, reside on some other computer across the LAN.

When you start up IPF, the GUI initializes itself, initializes `ipfsrv`, and then brings up IPF's window interface.

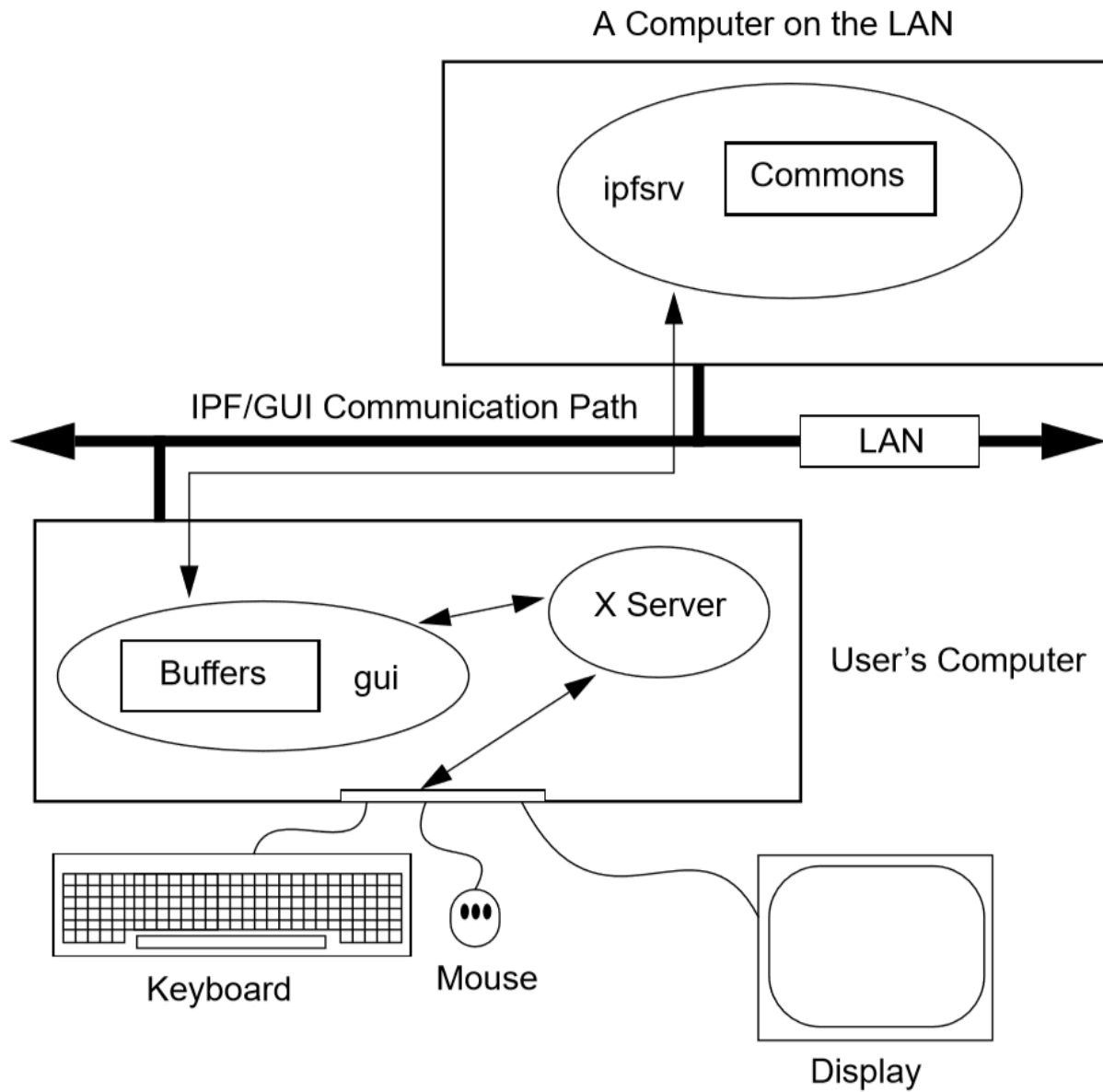


Fig. 2.10.4: PowerFlow/GUI Communication

### 2.10.5 IPF X Window GUI Architecture

The figure below shows a high-level view of the IPF X Window GUI and its environment. IPF is divided into the GUI code, which is written in C, and the powerflow code, which is written in Fortran. When you run the X Window System and Motif, you are using the Motif and X libraries in addition to Unix operating system calls and IPC (Interprocess Communication) calls. The GUI part of IPF (`gui`) uses the Motif library and IPC calls. The powerflow part of IPF (`ipfsrv`) uses IPC calls.

Because the Motif library is used for IPF, the Motif window manager is recommended for running with IPF. You can use another window manager with IPF, but the program will not have a consistent look and feel. The functionality remains the same, but you will notice a visual style within the IPF client that is different from the window manager, which controls the display outside the IPF windows and dialog boxes.

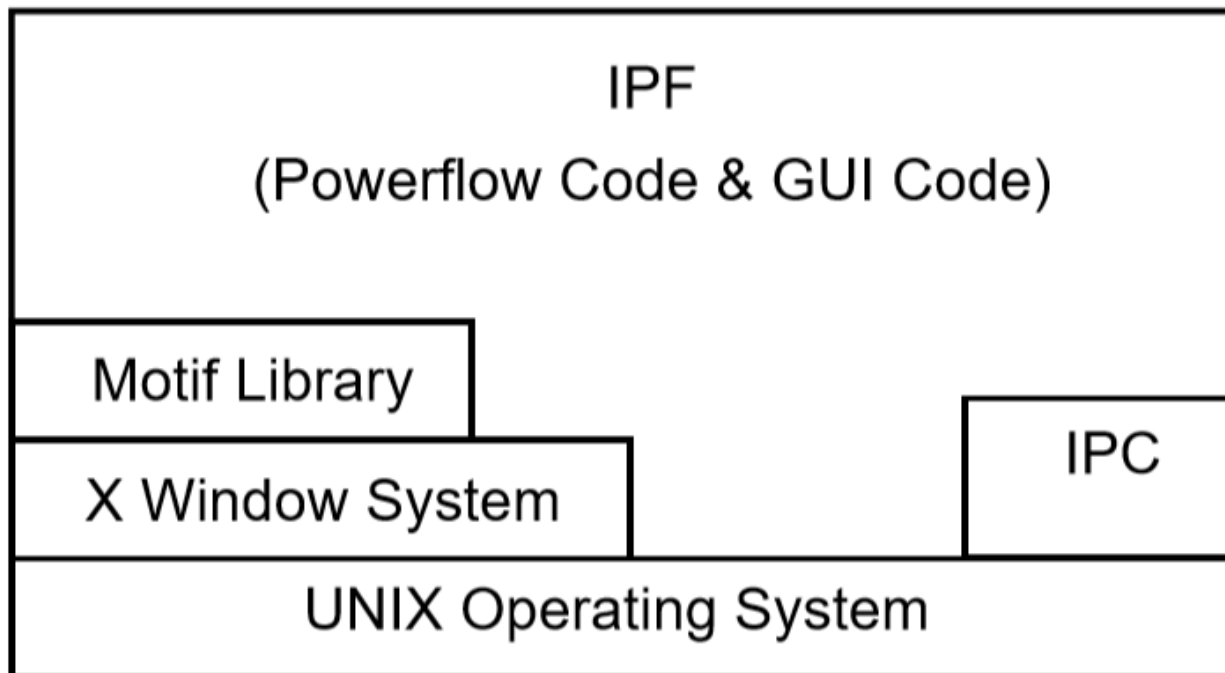


Fig. 2.10.5: Powerflow/GUI and System Software

### 2.10.6 History

Some key developments took place in the late 1970s and 1980s that made possible the X Window System and the X window managers.

- The idea of the graphical user interface (GUI) was created, implemented, and became popular.
- Personal computers became powerful enough to handle the data and processing intensive GUI.
- Efficient, fast, inter-computer communication - the local area network (LAN) — became widespread.
- A general purpose, non-proprietary operating system - the UNIX operating system — achieved wide acceptance.

All of these developments contributed materially to the possibility and, indeed, the eventual widespread acceptance of the X Window System and X window managers.

In the late 1970s, medium-sized computers began to get powerful enough to handle information organized graphically rather than in a character-based (textual) way. They also began to get cheap enough to be dedicated to one person. It was

on one of these computers that the Xerox Corporation installed the very first GUI using a mouse, menus, and windows. This GUI borrowed heavily from the Smalltalk-80 user interface, which was also invented at Xerox. (Smalltalk-80 is an object-oriented programming environment and system.)

GUI - graphical user interface - simply means using the graphics capability of a computer as the primary mode of interacting with users. A few GUIs do not, in fact, use a mouse, menus, and windows. However, GUIs using this combination of elements became common early because of their inherent ease of use.

Meanwhile, truly affordable personal computing was taking off in the late 1970s and early 1980s with the Apple and then IBM PC computers. These computer architectures were essentially graphical in nature, especially the Apple, though their interfaces were still character-based.

In 1984, Apple introduced the first personal computer with a thorough-going GUI. This was the Macintosh. This computer introduced wide numbers of people to a very easy to use graphical interface. It showed many computer manufacturers that they needed to design with GUIs in mind. A few years later, Microsoft Corporation retrofitted a windowing system onto MS DOS IBM PC-compatible computers.

Also in the late 1970s and 1980s, Sun Microsystems, Hewlett-Packard, DEC, and other companies saw the need for powerful computing “workstations” that business, research laboratories, and government could use. These workstations became common where personal computers didn’t have enough power. However, they generally lacked GUIs, which made them harder to use and less versatile than they might have been.

Another element in the computing picture was also developing in the 1970s and 1980s. This was widespread inter-computer communication. The idea of the local area networks (LAN), which was a room-to-room and building-to-building communication network, was created and implemented. A particularly popular LAN was, and is today, the Xerox-created Ethernet. This LAN is simply a cable connecting computers, whereby the computers can request and send just about any kind of data, often organized as files.

Another computing environment element was the development and distribution of a general purpose operating system that was platform independent, the UNIX system developed in the 1970s and 1980s at AT&T’s Bell Labs. This operating system was distributed freely to many college campuses, and the University of California at Berkeley developed many extensions to UNIX, among them sockets which provide efficient communication over LANs. Today, the UNIX operating system is offered commercially by AT&T as UNIX System V.

To complete the computer environment picture of the 1980s and 1990s, engineers at the Massachusetts Institute of Technology created the X Window System, often called just “X.” This GUI and underlying software was designed for powerful workstations with graphics-oriented hardware architecture. These capabilities are now available on high-end PCs as well. It is also designed for computers that make heavy use of a LAN. Though not limited to the UNIX operating system, the X Window System was first developed on UNIX computers.

### 2.10.7 Working with the GUI

This section describes how to accomplish basic tasks in the GUI version of IPF. After reading through this chapter, you should be able to use the Concepts and Commands section to figure out and work with the rest of IPF’s features.

The material in this section is not written as a complete, start-to-finish tutorial. Rather, the material is organized by individual task. However, the various topics are organized so that they reflect the common order of tasks in a typical session. So, you can either read the topics and do the steps within the topics in the order presented, or you can skip around and try out specific topics as they interest you.

---

**Note:** Most of the task procedures in this chapter involve using the mouse and its buttons. When you are asked to use a mouse button, the left-hand button is meant unless you are explicitly directed otherwise.

---

The main menus and toolbox are the keys to IPF’s main functions. The main menus contain commands that allow you to open files, save files, print network diagrams, solve cases, get help, edit bus and branch data, and so forth. The

toolbox contains icon buttons that move you into and out of various display modes. They allow you to create new buses, move buses, bend branches, reduce or enlarge the display, modify bus or branch data, and so forth.

The Display Mode buttons determine whether the map shows the intersection of the currently loaded base case file and the currently loaded coordinate file, or everything in the coordinated file. The current files area tells you which files you currently have loaded. The branch color key indicates the base kV rating of branches shown on the display, or their overload status.

In a prototypical session, you would generally follow this scenario:

1. Load a solved system and/or a network data file, and a coordinate file via the main menu Files - Open command.
2. Make changes to the system and/or coordinate file data using the various toolbox icon buttons, which allow direct on-screen manipulation of bus icons and branches, or bring up dialog boxes for adding data to bus and branch records.
3. Solve the new case with the main menu Process - Solve command.
4. Save your new case including its solution data with the main menu Files - Save command.
5. Print out a network diagram with the main menu Files - Print Plot command.

You might also do the following:

- Run a CFLOW program with the main menu Process - Auto CFLOW command.
- Get some help along the way with the main menu Help command.

### 2.10.8 Starting IPF

IPF is an X Window System application and is started like any other X application. However, your underlying operating system and the window manager you are running offer some ways to simplify how you start up IPF. Essentially, there are three ways you can start up IPF:

- You type the IPF command name `gui` in a terminal emulation window.
- You select IPF on your window manager's "run applications" menu. Generally, this involves clicking a mouse button on the background to bring up the "run applications" menu.
- You find that IPF is started automatically when you log in to your account. This means the system administrator has already set up your account to do this. You should see IPF as an icon or open window after the login process is complete.

To start IPF from an X terminal emulator window:

1. Make sure the X Window System and your window manager are running. Consult with your system administrator if you don't have X running.
2. Find a terminal emulator window or open one from a window manager menu.
3. In the terminal emulator window, enter `gui`. Within a minute or less, depending on the performance of your computer system, you should see the IPF main window appear.

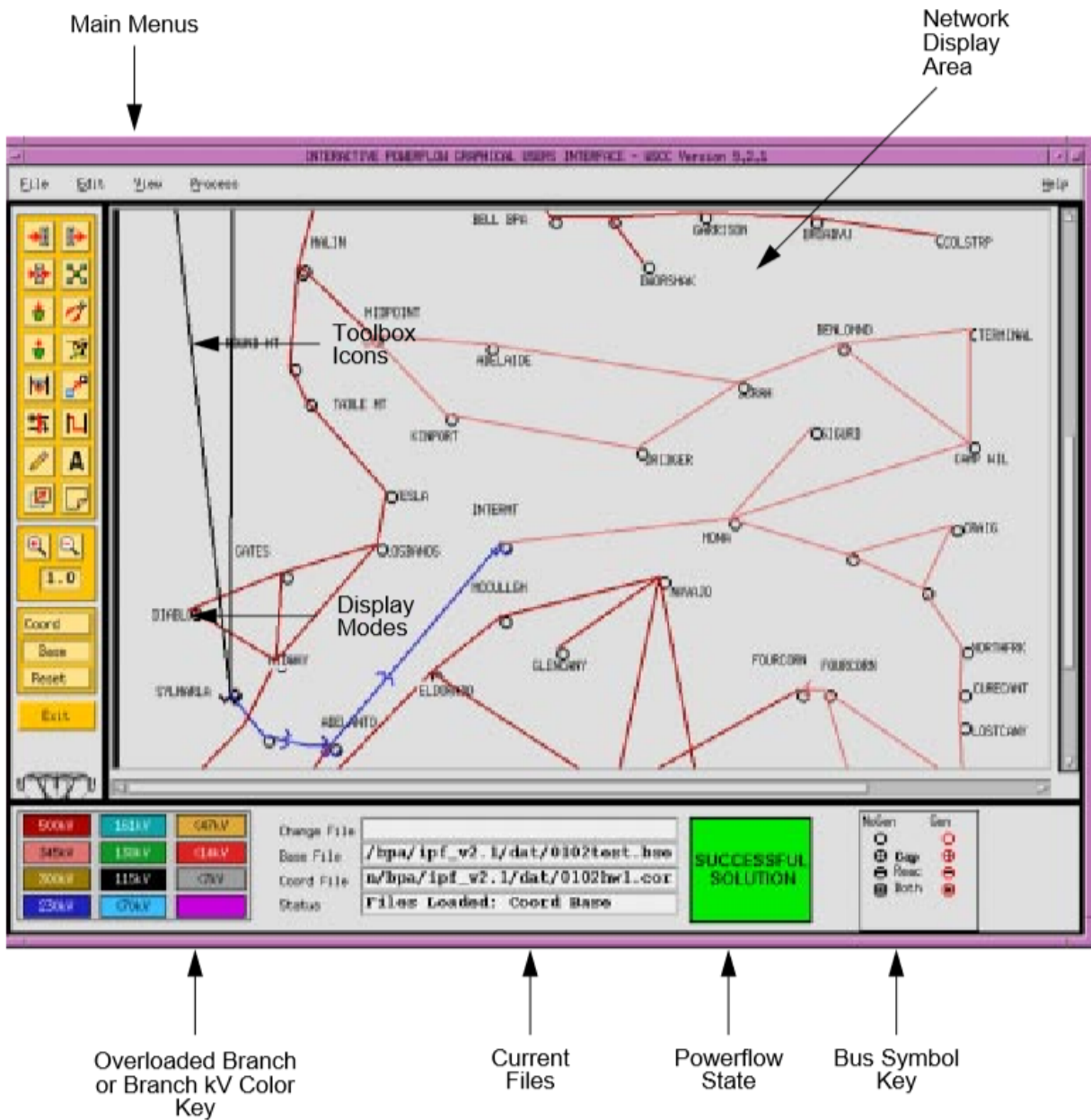


Fig. 2.10.6: Main Window

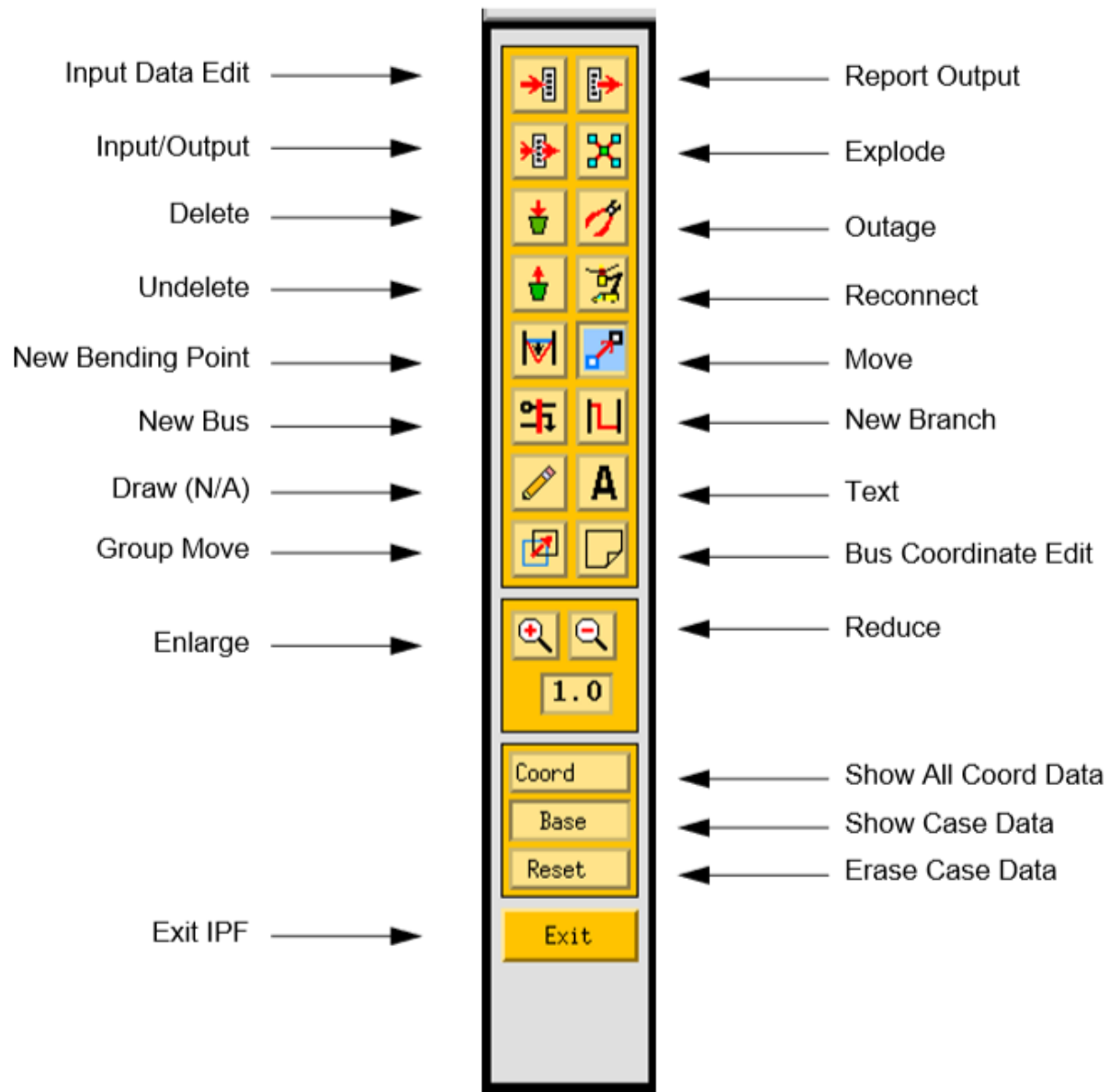


Fig. 2.10.7: Toolbox Icons

### 2.10.9 Exiting IPF

When you are through editing the currently loaded base case or coordinate files, running a solution, etc., and have saved your work, choose the Exit command from the File menu.

1. Click the File menu and select the Exit command. You will see the Exit dialog box come up. Do not use the window menu button (upper left) to close the IPF window.
2. Click OK if you are sure you want to exit IPF. Click Cancel if you have changed your mind and want to keep IPF running. If you select OK, IPF closes all its windows and removes them from the screen. You will still be in the X Window System, and you can proceed to any icon or open window to continue working with other X clients. If, after exiting IPF, you decide to run IPF again, see Starting IPF, above.

### 2.10.10 Opening Files

The Open dialog box shows you the five kinds of files you can open in IPF: the command, change, base case, network data, and coordinate files. See the figure below. For detailed information about these files, see *Powerflow Command Language (PCL)* and *Record Formats*.

For most power flow studies, the base case (binary) coordinate files are used. However, network data files must be used to initially create a binary base case file. Change files are used to make changes to a base case file; this case is called a change case, and the results are typically saved as a new base case file. Command files are *Powerflow Command Language (PCL)*.

Unlike the command, change, network data, and coordinate files, which are ASCII text files, the base case file is a binary memory image file. The base case file contains only power system data, which is edited within IPF, whereas the command, change, and network data files may be edited outside IPF using any ASCII text editor.

The coordinate file is a combination of bus position and branch bending point data in addition to plotting data. The coordinate file bus position and branch bending point locations can be altered by moving them in the GUI display, and a new coordinate file saved if desired. The plotting data can also be edited outside of IPF with an ASCII text editor.

#### Displaying a Network File

Normally, you will want to load a coordinate file in addition to the network data or base case file you intend to work on. If you do not, you will not be able to display the system graphically in the IPF main window. A system data file needs coordinate position information to display itself in IPF. It must get this information from a coordinate file since it does not have this information within itself. However, if all you want to do is edit a coordinate file, you can load just that file, and it will display properly in the main window.

If you load only a system data file, you can use only the textual editing and report capabilities of IPF to see your data, but you can still solve, make changes, save cases, etc. Also, you can generate a network diagram on the fly from which you can graphically navigate or display results. See *Exploring Base Case Connectivity*.

To open a network file:

1. Choose Open from the file menu in the IPF main window. You will see the Open dialog box as shown in Figure 3-3.
2. Find Network Data File at the left of the dialog box. The button, information box, and file text boxes under the heading allow you to select a network data file. The file text box under the Select button holds a file selection string. The string displayed when you first open the Open dialog box comes from a default that you can set in your XGUI file. See Appendix A, Customizing IPF.
3. Click the Select button. Notice that the file selection string is transferred to the Filter file text box at the right. Also, the Files list changed to reflect the Filter criteria.
4. Scroll the Files list to find the file you want to load.



5. Select the file by clicking it. This action puts the selected file in the Selection file text box below the Files list.
6. Click the Apply button to put the file name you have selected in the Network Data File file text box at the left side of the Open dialog box.
7. Click the Load Selections button at the bottom of the left side of the Open box. This final action actually loads the selected file into IPF's memory.

Since step 7 causes the open files dialog to close, it is best to perform steps 1 to 6 for each of the different kinds of files you want to load, and then click Load Selections at the end to load all files at one time. Note that you cannot open a network and a base file. Only the last one you picked will be loaded.

To open a command, change, base case, or coordinate file:

To open these files, perform the steps above. All of the essentials are similar; only the file type is different.

### 2.10.11 Saving Files

You can save five kinds of files in the X Window GUI: change files, base case files, network files, coordinate, and stability interface files. Ordinarily, you save files after you are done with a work session, but you can save a file at any time. Change, network, and coordinate files are saved in ASCII text format. Base case files are saved in binary format. Stability interface files can be saved in either binary or ASCII format. In a binary file, a memory image of base case data is written out to a file. See [Overview](#) for more information about these files.

To save a file:

1. Choose Save from the File menu in the main IPF window. You will see the Save dialog box as shown in the figure below.
2. Change the name of the file you are saving if you do not want to overwrite an existing file.
3. Click the appropriate Save button to save the desired file.
4. If the file you are trying to save has the same name as an existing file, you are presented with the Overwrite warning dialog box. You can choose Overwrite to complete the save or Cancel to cancel the save. Note only one base per file is allowed.

### 2.10.12 Changing the Displayed Network Size

You can use the X Window scroll bars on the right and bottom edges of the network display window to see a different chunk of the network diagram. The X Window GUI also includes an enlarge (and reduce) displayed network feature. Two buttons in the toolbox control this. X Window GUI has three displayed network size options. When you first load a base case or coordinate file, the middle size is chosen by IPF. If you find you would like to see more detail in a network diagram, you choose the Enlarge button. If, on the other hand, you would like to see the overall picture, you choose the Reduce button. The Enlarge button doubles the displayed network size, and the Reduce button halves the image size.

All other toolbox operations work no matter what the network size is, so you are free to work with the size that best suits the task at hand.

To alter the displayed network size:

1. Make sure you have a coordinate file loaded. See Opening Files to find out how to do this.
2. Click the Enlarge button in the toolbox in the main window. This magnifies the displayed network by two times. Note that you are now at the top magnification, 2.0.
3. Click the Reduce button. The displayed network returns to its original size (1.0), which is the size created by an Open operation. Click the Reduce button again. The displayed network size decreases by half. This is the smallest size, 0.5.

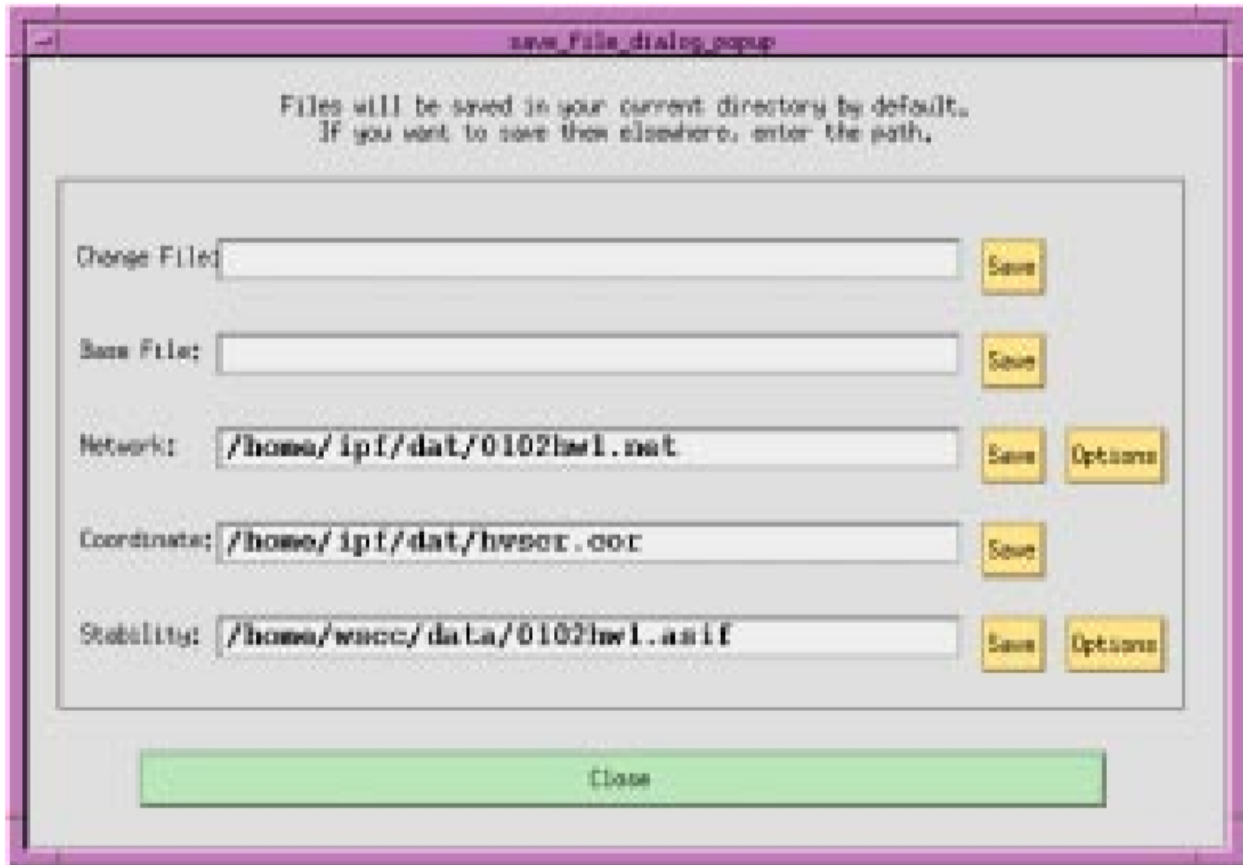


Fig. 2.10.8: Saving a File

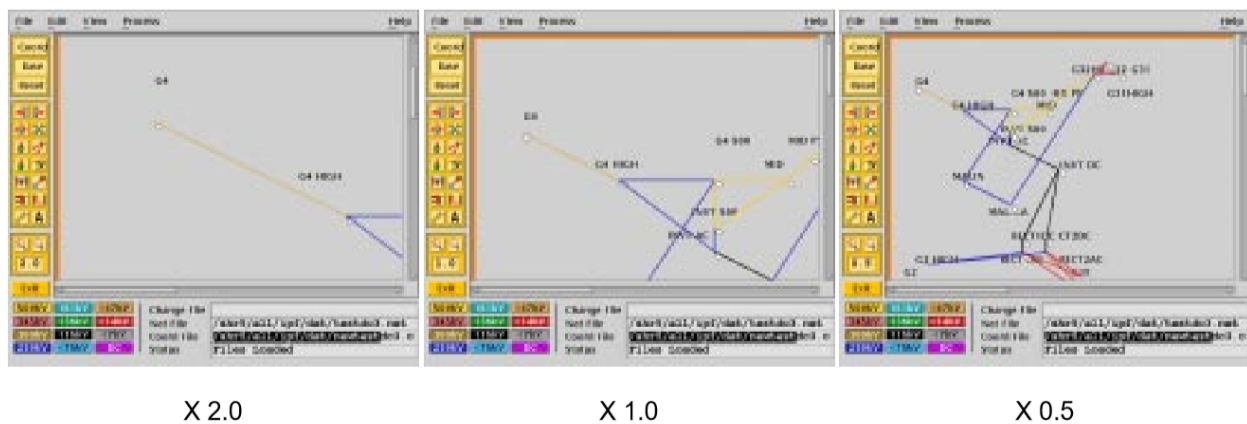


Fig. 2.10.9: Displayed Network Sizes

### 2.10.13 Editing Base Case Data

Editing a base case file is one of the primary activities in the X Window GUI. This involves a number of tasks: adding new buses or branches, modifying existing bus or branch values, and deleting existing buses or branches. These tasks are all accomplished in the main window toolbox and display area.

#### Adding a Bus and Related Components

You can add a bus to a resident base case or to IPF itself with no base case loaded. The procedure is the same for both. Adding a bus means in effect to create a new bus record within IPF's memory. You can add as many as you want up to IPF's maximum limit. Bus-related components are such items as continuation, transformer, line, and Q records. The method of adding these components is very similar to adding a bus record.

To add a bus:

1. Make sure you have a base case loaded if you want to add a bus to an existing base case file. See *Opening Files* to find out how to do this.
2. Click the New Bus icon in the toolbox in the main window. This puts you in the Add Bus mode and brings up the Bus Input Data dialog with a dummy bus name. Change this to whatever you want, and add data.
3. Move into the display area where you will note that the cursor changes to a box to remind you that you are in New Bus mode. Click anywhere in the display area. Note that a bus icon with the new name appears at the point where you clicked.

#### Modifying a Bus

Modifying a bus means to alter any of its associated values as they exist in the currently loaded base case data. This is done through the Input Data Edit Box. Refer to *Bus and Branch Editing* to find out more about the Input Data Edit Box.

To modify a bus:

1. Make sure you have system data loaded. See *Opening Files* to find out how to do this.
2. Click the Input Data Edit button in the toolbox. You are now in Input Data Edit mode.
3. Find a bus you want to modify, move the cursor over the bus, and click the left mouse button. The Input Data Edit Box appears with data text boxes filled with the values associated with the bus you clicked.
4. Change any of the values to new ones or type new values in blank text boxes. See the *Record Formats* for information on models and values. When you have values correctly typed (there is some data entry checking), click the Apply button at the bottom of the box to modify the bus data in the resident base case data.

#### Adding a Branch

You can add a branch to a resident base case. Adding a branch means to create a new connection between existing buses, which adds a new branch record within IPF's data structure. You can add as many as you want up to IPF's maximum limit. Branches are always associated with their terminating buses, so you access branches through buses.

To add a branch:

1. Make sure you have system data loaded. See *Opening Files* to find out how to do this.
2. Click the New Branch icon in the toolbox in the main window. This puts you in the Add Branch mode.

3. Move into the display area where you will note that the cursor changes to a right pointing arrow. Click the first bus that you want to connect the branch to. Move to the second bus that you want the branch to connect to and click on it. A line appears on the display connecting the two bus icons. Also, the Input Data Edit Box appears.
4. Fill in the values for the new branch as appropriate. See [Record Formats](#) for more information.
5. Click the Add button to add the new branch record to the currently resident base case data.

### Modifying a Branch

Modifying a branch means to alter any of its associated values as they exist in the currently loaded base case data. This is done through the Input Data Edit Box. Refer to [Bus and Branch Editing](#) to find out more about the Input Data Edit Box.

To modify a branch:

1. Make sure you have system data loaded. See [Opening Files](#) to find out how to do this.
2. Click the Input Data Edit button in the toolbox. You are now in Input Data Edit mode.
3. Find a bus that is connected to the branch you want to modify, move the cursor over the bus, and click the left mouse button. The Input Data Edit Box appears with data text boxes filled with the values associated with the bus you clicked. But you want a branch.
4. Find the option menu button (labeled “Bus”) at the upper right of the dialog box. This menu contains records associated with the currently selected bus. You will find continuation, transformer, branch, and other records on this menu.
5. Press this option button to show the associated items. Drag down to the branch record of your choice and release the mouse button. Note that the Input Data Edit Box now reflects data associated with the branch you chose.
6. Change any of the text box data to new values. When you have values correctly typed (there is some data entry checking), click the Apply button at the bottom of the box to modify the branch data in the resident base case data.

### Adding, Modifying, or Deleting an Area or Intertie

For area or intertie studies, you can add, modify, or delete areas or interties, by using the main window Edit - Area/Interchange command. You can do this at any time. See figures below.

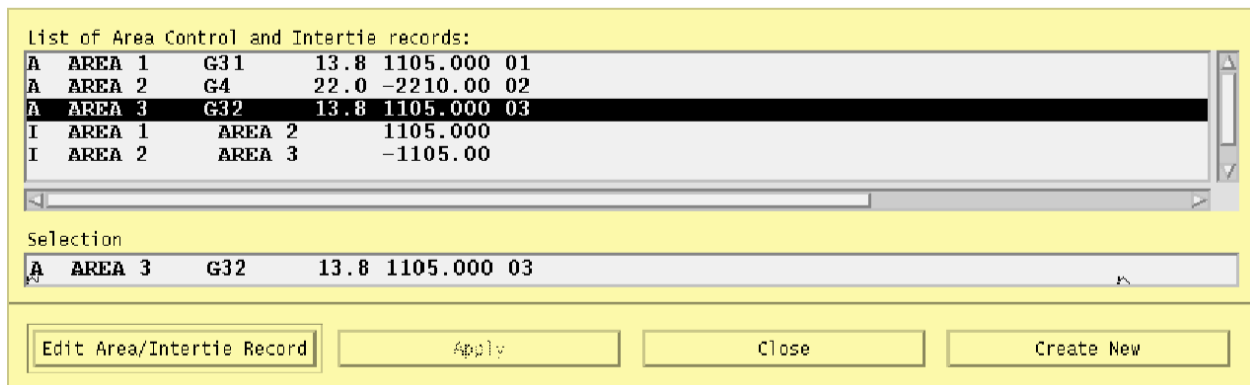
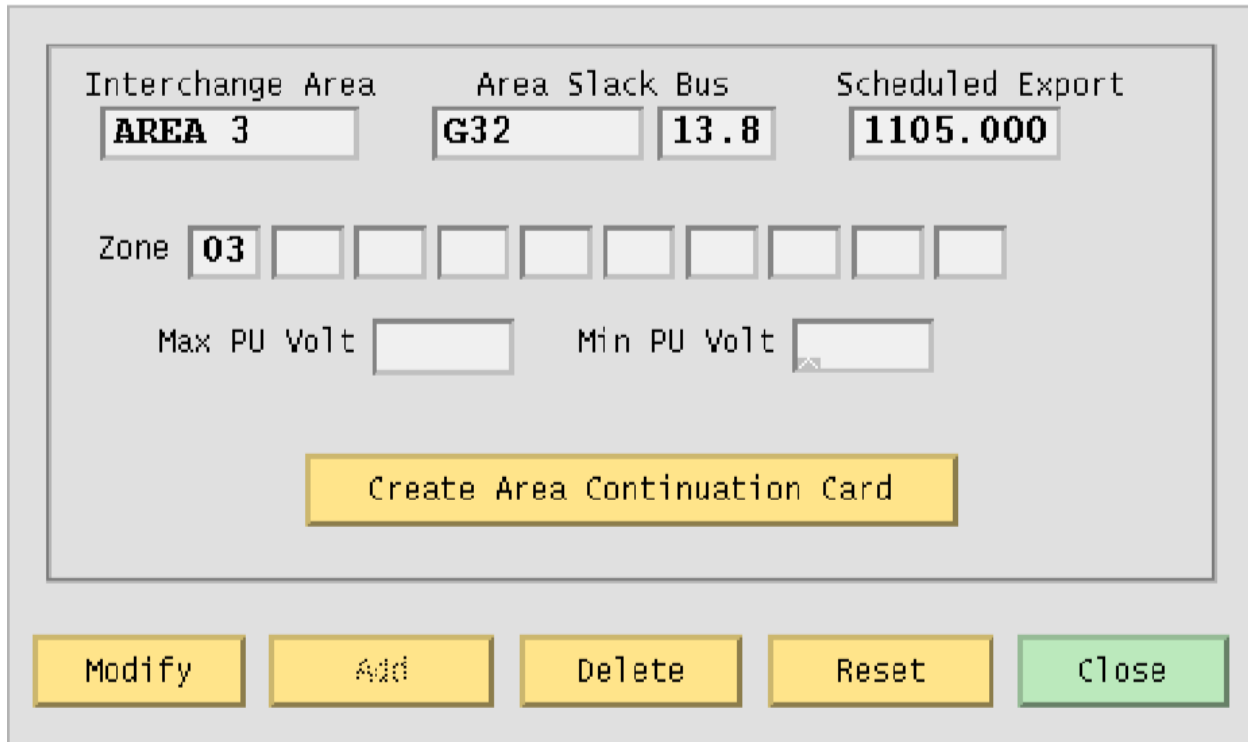


Fig. 2.10.10: Area Intertie Selection Dialog Box

To add an area:

1. Click Area/Interchange on the Edit menu in main window. The Area/Intertie Selection dialog box appears.

2. Click the Create New button. A small box appears, asking whether you want to create an Area Control (A) record or an Intertie (I) record. Click Area Control.
3. The Area/Interchange dialog box appears. Fill in the required text boxes.
4. Click the Add button at the bottom. The Area/Interchange dialog box closes and the new data you typed into the Area/Intertie dialog box now appears in the Area/Intertie Selection dialog box.



The screenshot shows the 'Area Interchange Dialog Box' with the following fields and controls:

- Interchange Area:** Text box containing 'AREA 3'.
- Area Slack Bus:** Text box containing 'G32'.
- Scheduled Export:** Text box containing '13.8'.
- Zone:** A row of ten text boxes, the first containing '03' and the others empty.
- Max PU Volt:** Text box.
- Min PU Volt:** Text box.
- Create Area Continuation Card:** A yellow button.
- Bottom Buttons:** A row of five buttons: 'Modify' (yellow), 'Add' (yellow), 'Delete' (yellow), 'Reset' (yellow), and 'Close' (green).

Fig. 2.10.11: Area Interchange Dialog Box

To modify an area intertie:

1. Click Area/Interchange on the Edit menu in the main window. The Area/Intertie Selection dialog box appears.
2. Select an item in the list window by clicking it. This action puts the selected item in the Selection text box.
3. Click the Edit Area/Intertie button. The Area/Interchange dialog box appears.
4. Change the data in the text boxes.
5. Click the Modify button. The Area/Interchange dialog box closes and the new data you typed into the Area/Intertie dialog box now appears in the Area/Intertie Selection dialog box.

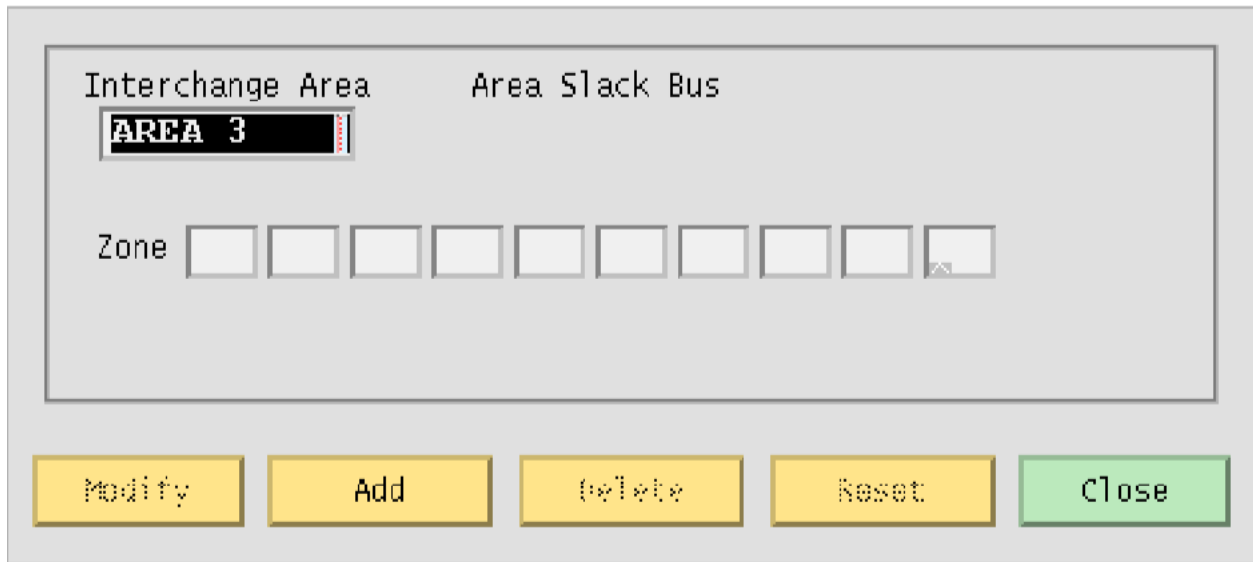
To delete an area:

1. Click Area/Interchange on the Edit menu in main window. The Area/Intertie Selection dialog box appears.
2. Select an item in the list window by clicking it. This action puts the selected item in the Selection text box.
3. Click the Edit Area/Intertie button. The Area/Interchange dialog box appears.
4. Click the Delete button.

To create an area continuation record:

1. Click Area/Interchange on the Edit menu in the main window. The Area/Intertie Selection dialog box appears.

2. Select an item in the list window by clicking it. This action puts the selected item in the Selection text box.
3. Click the Edit Area/Intertie button. The Area/Interchange dialog box appears.
4. Change the data in the text boxes if you need to.
5. Click the Create Area Continuation Card button. The dialog box appears.
6. Add zone data to the Zone text boxes.
7. Click the Add button. The Area/Interchange dialog box closes and the new data you typed into the Area/Intertie dialog box now appears in the Area/Intertie Selection dialog box.



The dialog box is titled "Area Continuation Card Dialog Box". It contains two main sections. The top section has two labels: "Interchange Area" and "Area Slack Bus". Below "Interchange Area" is a text box containing "AREA 3". Below "Area Slack Bus" is a text box that is empty. The bottom section is labeled "Zone" and contains ten small square boxes for entering zone data. Below the "Zone" section are five buttons: "Modify", "Add", "Delete", "Reset", and "Close". The "Close" button is green, while the others are yellow.

Fig. 2.10.12: Area Continuation Card Dialog Box

### Exploring Base Case Connectivity

Sometimes you may want to explode a portion of a large network to see how buses are locally interconnected. The idea is that you start with one bus on the display and find out what other buses are connected to it. Then, with each of these buses, you find out what buses are connected to these, and so on.

The Explode icon in the toolbox allows you to explore base case connectivity.

To explore a base case:

1. Load just a base case file to demonstrate this function. See [Opening Files](#) to find out how to do this.
2. Click the New Bus icon in the toolbox to enter Add Bus mode. The Input Data Edit box will come up; just close it without entering any data.
3. Select Alpha Search on the View menu. In the Alpha Search dialog box, type the first few letters of a bus name. The alphabetical list automatically scrolls to find the bus of interest in the list. See the Alpha Search command entry in Chapter 4 for more information.
4. Click the bus name of interest. It may already be highlighted, but you must click on it to make it the currently selected bus.
5. Move to the blank display area. (You should still be in Add Bus mode.) Click once to make the bus icon and name appear.

6. Move back to the toolbox and click the Explode icon. You previously established a bus that you can now explore bus connectivity with.
7. Click the bus. You will note that any buses that are connected to your bus of interest are now shown on the display with connecting lines representing branches. You can continue exploring the network by clicking any new buses that show up. You can reveal the entire network in this way if you like (although it will probably look like a mess!)

The buses and branches are positioned by an internal algorithm since you have not loaded any underlying coordinate data. You can click the Move icon and then move the buses around the screen if you want to clean things up.

## Sectionalizing a Bus

Sectionalizing a bus separates a bus into two buses and rearranges its branches between the two buses. You can also optionally create a tie line between the two buses. See figure below.

To sectionalize a bus:

1. Make sure you have system data loaded. See *Opening Files* to find out how to do this.
2. Click the Input Data Edit mode button in the toolbox. You are now in Input Data Edit mode.
3. Select a bus by clicking it. This brings up the Input Data Edit Box. At the center bottom of the dialog box is the Sectionalize button. Click it to cause the Bus Sectionalize dialog box to appear. Note that the bus name of the currently selected bus appears in two places.
4. Type a new bus name over the existing Bus 2 name to create a new bus.
5. Click any branch, transformer, etc., record in the left-hand scrolling text box to transfer it to the right-hand scrolling text box. Note that you can go back and forth by clicking the appropriate records till branches, transformers, etc., are all associated with the bus you want.
6. Click the OK button to send the sectionalized bus data to the resident base case data. If IPF detects any errors or inconsistencies, it puts up the Error dialog box. Examination of the message helps you figure out the problem.

---

**Note:** If the name you type is not accepted, IPF has found it to be a duplicate name and rejects it. If, at any point, you would like to start from the beginning, just press Reset at any time. This returns all values to the state they were at the time the dialog box first opened.

---

7. Sometimes you may want to connect the old and new buses making up the sectionalized bus. Click on the Bus Tie button to create a line with impedance of  $0.0 + j0.00001$ . You may modify this line later, if you wish

## Tapping a Line

Tapping a line means to service a new load by creating a new tap point bus on an existing line. The tapped line is effectively segmented into two lines, separated with a newly created bus. If the load is remote from the tapped point, an additional line and bus will be necessary. The new load and the new bus are connected by a new line. As with many line operations, you access line tapping through a bus that the line is connected to.

To tap a line:

1. Make sure you have system data loaded. See *Opening Files* to find out how to do this.
2. Click the Input Data Edit mode button in the toolbox.
3. Click a bus icon that is connected to the line you want to tap. This brings up the Input Data Edit Box which is loaded with input data pertaining to the selected bus. Click and hold the option button opposite the bus name at the top of the Input Data Edit Box. The cascading menu lists all branches connected to the current bus. Drag

Sectionalized Bus 1

**B**    **G2 HIGH 23002**

L	G2 HIGH 230	RECT1AC	23
L	G2 HIGH 230	RECT2AC	23

Sectionalized Bus 2

**TEST**

T	TEST GH 230	G2	13.
---	-------------	----	-----

After Renaming Bus 2, Click on item to transfer to other bus

Bus Tie

OK    Reset    Close    Help

Fig. 2.10.13: Sectionalize Operation Completed

down to the line you are interested in tapping. Be sure that this is a line and not a transformer. Selecting this item brings up the Input Data Edit Box for your line of interest. Near the center bottom of this dialog box is the Tap Line button. Click it to cause the Tap Line dialog box to appear.

4. Initially the dialog box displays data from a previous invocation. If this is the first time it is displayed, all fields are blank. Enter the name of the bus you want to create in the dialog field Tapped Bus Name and press the Apply button. The selected line is tapped at the point depicting 50% of the total line's reactance. You can horizontally scroll the line data within the Bus 1 side and the Bus 2 side to verify that the line resistance and reactance is split according to the sliding scale selected. Pressing the Apply button recomputes the line impedance on each side of the tapped bus without affecting any data in the Powerflow base case.
5. Change the units (Percentage, Miles, Kilometers, or Section) to reflect your tapping criteria. Adjust the horizontal slider as necessary. Move the slider by moving the mouse cursor over it, pressing the left mouse button, and moving left or right till you find the point of the line you want to tap. Release the mouse button. The line tapping slider just above the Reverse Scale button shows the tap point according to the tapping criteria (length of the line in kilometers or miles, or percent of reactance). The line tapping slider also shows any sections the line may be divided into. Again press Apply to update the line's impedance to reflect the slider's current value.
6. If the values are acceptable, click OK to export these changes to Powerflow.
7. Click Close to cause the dialog box to disappear.



### 2.10.14 Solving a Network Case

Solving a network case (or base case) causes IPF to calculate bus voltages that satisfy the network constraints as they exist within the currently resident base case data. This is usually done after you have loaded a base case and made some modifications to reflect the conditions of the system you want to study. However, you may solve as soon as you have loaded any system data. You do not need to load a coordinate or change file to solve a case. See figure below for a typical display after a case solution.

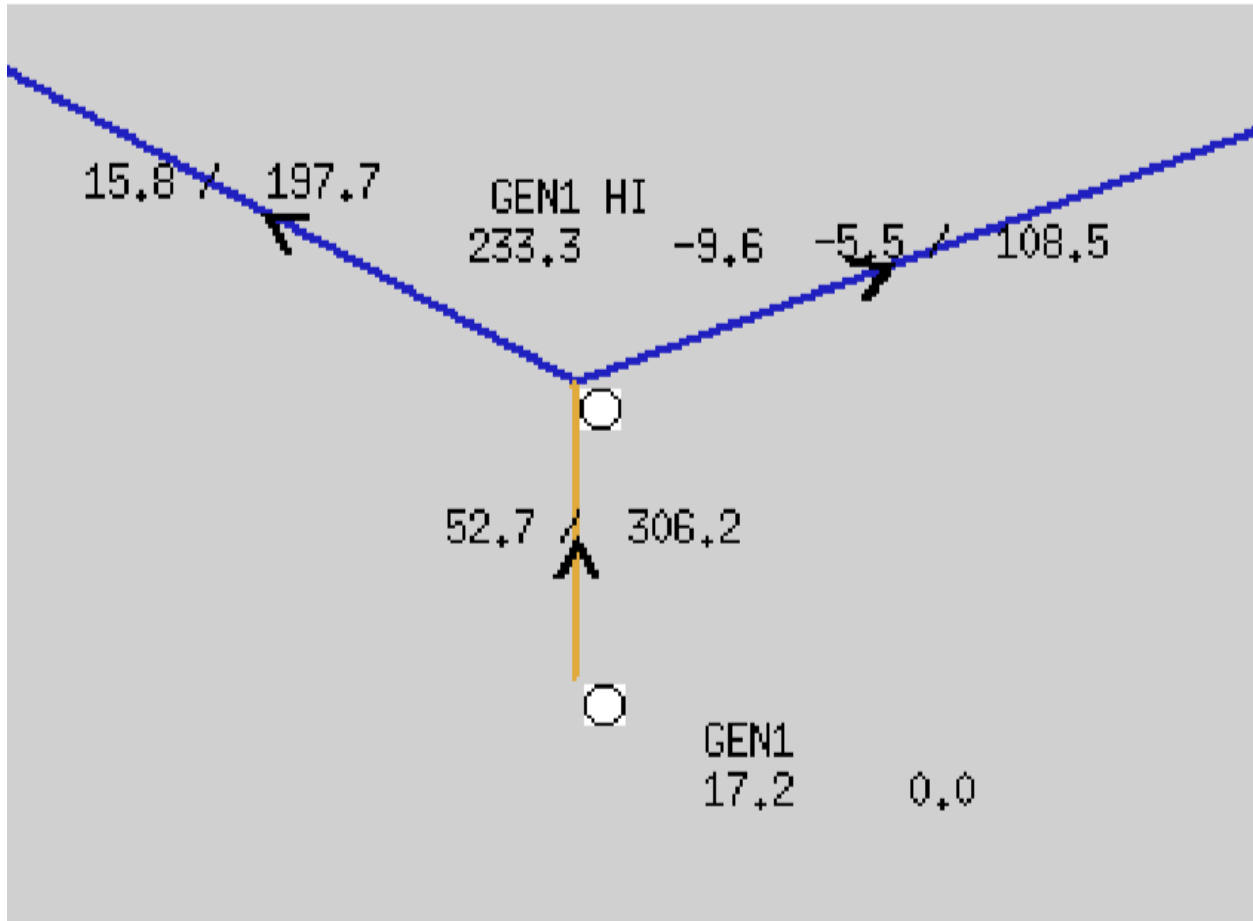


Fig. 2.10.14: Network Display After Solution

To solve a base case:

1. Make sure that you have previously loaded a base case or netdata file. See [Opening Files](#) if you have not.
2. Choose Solve Case from the Process menu in the IPF main window. The Solve dialog box opens and you are presented with a number of options. See the figure below. Note that default values are set for you. If these suit your case, click the Solve button. If the defaults are not appropriate for your case, change them. Then solve the case. See Chapter 4 for a discussion of the options.

The solution may take anywhere from a few seconds to minutes depending on the number of buses in the base case. After the solution is complete, the display shows some of the calculated data.

### 2.10.15 Bus and Branch Editing

Bus and branch data editing are most often accomplished through the Input Data Edit Box. This dialog box is brought up on the display whenever you click a bus displayed in the main window. The specific data associated with the bus you click fills the text boxes of the Input Data Edit Box. You can change any of the values to new values. You can then store the changed data in the memory-resident bus and branch database, solve the case with the new data values, and optionally save the changes permanently in a file.

---

**Note:** See the Network Data Edit menu item entry in this chapter for another way to edit bus and branch data.

---

Each of the text box fields in a given instance of the Input Data Edit Box corresponds to a record field for a bus, line, or transformer type as documented in the *Record Formats* section. If you are unsure of the meaning of the fields for a particular bus, line, or transformer type, look it up in that section.

In the *Record Formats* section, field width, decimal point placement, and sign are specified for each field. Real number fields have a position where a decimal point is implied, i.e. as long as you correctly position your digits, you need not enter a decimal point in a .pfc file or NETWORK\_DATA file. However, in the Input Data Edit Box you should always enter a decimal point. Sufficient room has been allowed for this in each field. The GUI will format your data correctly so it fits the field on a standard input record.

When you enter data into the Input Data Edit Box text boxes, the GUI checks to see whether you have entered values correctly. Specifically, it checks for all characters being numeric, having a minus sign (in the proper place), or having a decimal point (period). All other characters are rejected and a warning bell sounds. Other basic checks for valid data are also operating in during data entry. However, it may still be possible to enter invalid data that only shows up when a power flow solution is attempted.

#### Bus Editing

Bus input data dialogs are accessed directly by clicking on a bus icon in the display, or by selecting a bus name on the Alpha List. Their appearance varies according to the type of the bus. To change the bus type of a bus, click the button labeled Type, near the Owner and Zone fields. This option button pops up a menu showing all the bus types accepted by IPF. If you click on a different type than was originally displayed, the dialog will change into one appropriate to that bus type. If you click the Modify button, the bus will be changed to a different type. You may have to supply more (or different) data in other fields in order for the change to be legal.

#### Branch and Other Component Editing

You cannot select a branch directly. Branch data is accessed by selecting a bus that is connected to the line or transformer you want to edit. Then, once the Input Data Edit Box shows a bus connected to your line of interest, you pick the line by using the option button labeled Bus, in the upper right corner of the box. Press and drag down to select the one you want. The dialog box changes to reflect the fields and parameters characteristic of lines.

The Bus option menu also contains other bus-related components, or records, such as switched reactance (X) data, bus continuation data, PQ curves, etc. Drag down and release to select any of these you want to edit.

## Adding New Components

New buses are added by using the New Bus tool in the main menu toolbox. See *Adding a Bus and Related Components* for details on this process. New branches can be added only if both terminal buses exist. You can add branches graphically by using the New Branch tool, or by going through the Input Data Edit Box for one of the terminal buses. Other bus-related components can only be added through the dialog box. To add components to a bus, press the option button labeled New Component and drag down to the desired item. Items which are grayed out are not appropriate for this type of bus. For example, switched reactance (X) data can only be added to a BX type bus.

When you release the mouse button, the data dialog for the selected item will appear. You fill in the desired data fields, and click the Add button at the bottom to add it to the currently resident base case data. Notice that the Modify button is grayed out on this dialog.

### 2.10.16 Input Data Edit Dialog Boxes

The following dialog boxes are all accessed from the Input Data Edit Dialog Box. The dialog boxes are arranged with buses first and branch components last.

Table 2.10.1: Input Data Edit Dialog Boxes

Dialog	Box Description
B-BLANK BUS	Adds bus data for modeling load bus.
BC BUS	Adds bus data for a bus controlled by a BG bus.
BD BUS	Supplies data for a two terminal dc bus.
BE BUS	Adds bus data for a bus that holds its voltage to a specified value.
BF BUS	Supplies data for a special-purpose bus for the NewtonRaphson solution method.
BG BUS	Adds bus data for a bus that controls the voltage of a remote BC bus.
BM BUS	Supplies data for a multi-terminal dc bus.
BQ BUS	Adds bus data for a bus that holds its voltage to a specified value within reactive limits.
BS BUS	Adds bus data for the slack (or swing) bus.
BT BUS	Adds bus data for a bus that maintains its voltage with an LTC transformer.
BV BUS	Adds bus data for a bus that holds its net reactive power within a user-specified voltage range.
BX BUS	Adds bus data for a bus that controls its local voltage by switching capacitors/reactors in and out.
CONTINUATION BUS	Adds additional data to an existing bus record.
SWITCHED REACTANCE	Adds data for voltage controlled shunt device installations.
PQ CURVE	Adds PQ curve data for calculation of Q limits.
SECTIONALIZATION	Provides for the sectionalization of a bus.
LINE TAPPING	(Not yet available) Provides for tapping of lines.
TRANSMISSION LINE	Adds data for a balanced transmission line.
PHASE SHIFTER	Adds data for phase shifting transformers.
TRANSFORMER	Adds data for two-winding transformers.
REGULATING TRANSFORMER	Adds data to give fixed transformers or phase shifters automatic regulating or control status.
EQUIVALENT NETWORK	Adds data for an asymmetrical pi type line.

### 2.10.17 AC Bus Input Data Boxes

This box will look slightly different, depending on the bus type. All of the AC bus data fields are described below; specific differences are covered under the various bus types. You must click separately in each field that you want to alter. Text entry is always in Insert mode, meaning that you cannot type over a character or number; you must delete it first. Note that when a field is described as a “real number,” you should *always* enter a decimal point. See below for an example of a typical bus input data dialog box.

**Name.** An eight character maximum, alphanumeric string designating a bus name. The string must start with an alpha character. The name should be unique. This name is placed in the first text box from the left. The right text box should have a five character maximum numeric string representing the bus’s base kV rating.

**Bus.** An option button that allows you access to other records that are associated with the currently displayed bus. Press on this button and drag down to select an existing branch or other component. The dialog box transforms into the dialog for the selected component.

**Owner.** Three character maximum, alphanumeric string designating a bus owner.

**Zone.** Two character maximum, alphanumeric string designating the zone the bus is in.

**Type.** An option button that changes the dialog box to reflect different bus types. The types are: B `` (B-blank), ``BC, BE, BF, BG, BQ, BS, BT, BV, and BX. You cannot change an AC bus into a DC bus using this button.

**Load P.** Five character maximum, real number designating real load in megawatts (MW).

**Load Q.** Five character maximum, real number designating reactive load in megavoltamperes reactive (MVAR).

**Shunt P.** Four character maximum, real number designating the shunt admittance load in megawatts (MW) at the base kV of the bus.

**Shunt Q.** Four character maximum, real number designating shunt admittance in megavoltamperes reactive (MVAR). A positive value is capacitive; a negative value is inductive. The minus sign goes in front of the number.

**P Max.** Four character maximum, real number designating the maximum real power generation in megawatts (MW).

**P Gen.** Five character maximum, real number designating scheduled real power generation in megawatts (MW).

**Q Sched.** Five character maximum, real number designating scheduled reactive power in megavoltamperes reactive (MVAR). May be positive or negative.

**Q Max.** Five character maximum, real number designating maximum reactive power in megavoltamperes reactive (MVAR). Generally positive.

**Q Min.** Five character maximum, real number designating minimum reactive power in megavoltamperes reactive (MVAR). Generally negative. The minus sign goes in front of the number.

**V Hold.** Four character maximum, real number designating a voltage to hold for the bus, in per unit.

**V Max.** Four character maximum, real number designating a maximum voltage limit in per unit.

**V Min.** Four character maximum, real number designating a minimum voltage limit in per unit.

**Sectionalize.** A button that brings up the Sectionalize dialog box. See Sectionalize Dialog Box in this section.

**New Component.** A tag for the option button that allows you to add a new bus-related component such as a branch, X data, etc. **Add.** A button that adds a new record to the database.


**Modify.** A button that modifies the record.

**Reset.** A button that restores text box displays to their original values (before any changes were made).

**Delete.** A button that deletes (removes) a record from the database.

**Outage.** (Not yet implemented.)

Name



Owner  Zone

Type

Load P  Load Q  Shunt P  Shunt Q

P Max  P Gen  Q Sched  Q Min

V Max  V Min

Q | Constant Q  
←  
0

New Component

Fig. 2.10.15: Input Data Edit Box Showing B-Blank Bus Data.png  
2.10. X Window Graphical Interface (gui)

**Close.** A button that causes the dialog box to close and disappear from the display without making any modifications to the record.

### B-Blank Bus

The B-blank dialog box supplies data for modeling the typical load bus. See the B-blank record in *AC Bus Data (B-blank)*.

**V Max.** Four character maximum, real number designating a maximum voltage limit in per unit.

**V Min.** Four character maximum, real number designating a minimum voltage limit in per unit.

The voltage limit fields take effect only if the voltage of the B-blank bus is being controlled by a remote device.

### BC Bus

The BC dialog box supplies data for a bus controlled by one or more BG type buses. See the BC record in *AC Bus Data (BC)*.

**V Hold.** Four character maximum, real number designating a voltage to hold for the bus, in per unit.

### BE Bus

The BE dialog box supplies data for a bus that holds its voltage to a specified value. See the BE record in *AC Bus Data (BE)*.

**Q Max.** Five character maximum, real number designating maximum reactive power in megavoltamperes reactive (MVAR). Generally positive.

**Q Min.** Five character maximum, real number designating minimum reactive power in megavoltamperes reactive (MVAR). Generally negative. The minus sign goes in front of the number.

**V Hold.** Four character maximum, real number designating a voltage to hold for the bus, in per unit.

### BF Bus

The BF dialog box supplies data for a special-purpose bus for the Newton-Raphson solution method. It holds the specified voltage until the P-solution has converged, then acts like an ordinary load bus (B-blank). See the BF record in *AC Bus Data (BF)*.

**Q Max.** Five character maximum, real number designating maximum reactive power in megavoltamperes reactive (Mvar). Generally positive.

**Q Min.** Five character maximum, real number designating minimum reactive power in megavoltamperes reactive (Mvar). Generally negative. The minus sign goes in front of the number.

**V Hold.** Four character maximum, real number designating a voltage to hold for the bus, in per unit.

### BG Bus

The BG dialog box supplies data for a bus that controls the voltage of a remote BC bus. See the BG record in *ac-bus-data-bg*.

**Q Max.** Five character maximum, real number designating maximum reactive power in megavoltamperes reactive (MVAR). Generally positive.

**Q Min.** Five character maximum, real number designating minimum reactive power in megavoltamperes reactive (MVAR). Generally negative. The minus sign goes in front of the number.

**V Max.** Four character maximum, real number designating a maximum voltage limit in per unit.

**V Min.** Four character maximum, real number designating a minimum voltage limit in per unit

**Remote Bus.** An eight character maximum, alphanumeric string designating the remote bus to be voltage controlled (a BC type bus).

**PCS.** A three character maximum, numeric string designating the percentage of VARS supplied by this bus to control the remote bus voltage.

### BQ Bus

The BQ dialog box supplies data for a bus that holds its voltage to a specified value within reactive limits. See the BQ record in *AC Bus Data (BQ)*.

**Q Max.** Five character maximum, real number designating maximum reactive power in megavoltamperes reactive (MVAR). Generally positive.

**Q Min.** Five character maximum, real number designating minimum reactive power in megavoltamperes reactive (MVAR). Generally negative. The minus sign goes in front of the number.

**V Hold.** Four character maximum, real number designating a voltage to hold for the bus, in per unit.

### BS Bus

The BS dialog box supplies data for the system slack (or swing) bus. See the BS record in *AC Bus Data (BS)*.

**Q Sched.** Five character maximum, real number designating scheduled reactive power in megavoltamperes reactive (MVAR). May be positive or negative.

**V Hold.** Four character maximum, real number designating a voltage to hold for the bus, in per unit.

**Angle.** Four character maximum, real number designating a voltage phase angle in degrees. Blank is translated to an angle of zero.

### BT Bus

The BT dialog box supplies data for a bus that maintains its voltage with an LTC transformer. See the BT record in *AC Bus Data (BT)*.

**Q Sched.** Five character maximum, real number designating scheduled reactive power in megavoltamperes reactive (MVAR). May be positive or negative.

**V Hold.** Four character maximum, real number designating a voltage to hold for the bus, in per unit.

### BV Bus

The BV dialog box supplies data for a bus that holds its net reactive power within a user-specified voltage range. See the BV record in *AC Bus Data (BV)*.

**Q Sched.** Five character maximum, real number designating scheduled reactive power in megavoltamperes reactive (MVAR). May be positive or negative.

**V Max.** Four character maximum, real number designating a maximum voltage limit in per unit.

**V Min.** Four character maximum, real number designating a minimum voltage limit in per unit.

### BX Bus

The BX dialog box supplies data for a bus that controls its own or a remote bus's voltage by switching capacitors or reactors in and out. See the BX record in *AC Bus Data (BX)*.

**Q Max.** Five character maximum, real number designating maximum reactive power in megavoltamperes reactive (MVAR). Generally positive.

**Q Min.** Five character maximum, real number designating minimum reactive power in megavoltamperes reactive (MVAR). Generally negative. The minus sign goes in front of the number.

**V Max.** Four character maximum, real number designating a maximum voltage limit in per unit.

**V Min.** Four character maximum, real number designating a minimum voltage limit in per unit.

**Remote Bus.** An eight character maximum, alphanumeric string designating the remote bus to be voltage controlled.

### 2.10.18 BD Bus

The BD dialog box supplies data for a two-terminal DC bus. See the BD record two-terminal-dc-bus-data-bd.

**Number of Bridges.** Two digit integer designating the number of bridges per dc circuit (number of valves serially connected).

**Smoothing Reactor.** Five character maximum, real number designating smoothing inductance in millihenries.

**Min Firing Angle.** Five character maximum, real number designating minimum firing angle ( $\alpha_{min}$ ) in degrees, for rectifier operation.

**Max Firing Angle.** Five character maximum, real number designating maximum firing angle ( $\alpha_{stop}$ ) in degrees, for inverter operation.

**Valve Drop.** Five character maximum, real number designating valve voltage drop per bridge, in volts.

**Bridge Rating.** Five character maximum, real number designating maximum bridge current rating in amps.

**Commutating Bus.** Eight character maximum, alphanumeric string designating the commutating bus name. This is the bus on the ac system side of the commutating transformer bank.



Name

Owner  Zone  Type D

Number of Bridges

Smoothing Reactor  mh

Min Firing Angle  Degrees

Max Firing Angle  Degrees

Value Drop  Volts

Bridge Rating  Amps

Commutating Bus

Add Modify Reset

Delete Outage Close

Fig. 2.10.16: Input Data Edit Box Showing BD Bus

### 2.10.19 BM Bus

The BM dialog box supplies data for a multi-terminal dc bus. See the BM record [Multi-Terminal DC Bus \(BM\)](#).

**Number of Bridges.** Two digit integer designating the number of bridges per dc circuit (number of converters serially connected).

**Smoothing Reactor.** Five character maximum, real number designating smoothing inductance in millihenries.

**Min Firing Angle.** Five character maximum, real number designating minimum ignition delay angle ( $\alpha_{min}$ ) in degrees.

**Max Firing Angle.** Five character maximum, real number designating maximum ignition delay angle ( $\alpha_{stop}$ ) in degrees.

**Valve Drop.** Five character maximum, real number designating converter valve drop per bridge, in volts.

**Bridge Rating.** Five character maximum, real number designating bridge current rating (maximum converter current) in amps.

**Commutating Bus.** Eight character maximum, alphanumeric string designating the commutating bus name.

**Converter Type.** Single character alpha string designating the converter code. R indicates normal operation as a rectifier; I is normal operation as an inverter. M indicates an inverter with current margin, and blank indicates a passive dc tap.

**Ignition Delay Angle.** Three character maximum, real number designating the normal ignition delay angle ( $\alpha_N$ ) for a rectifier, or normal extinction angle ( $\gamma_N$ ) for an inverter, in degrees.

**Min Extinction Angle.** Three character maximum, real number designating the minimum ignition angle ( $\alpha_{min}$ ) for a rectifier, or minimum extinction angle ( $\gamma_0$ ) for an inverter, in degrees.

**Converter DC Power.** Six character maximum, real number designating the scheduled dc bus load (net converter dc output power) in megawatts (MW) at the base kV of the bus.

**Converter DC Voltage.** Five character maximum, real number designating the scheduled dc bus kV (converter dc voltage).

### 2.10.20 Continuation Bus

The continuation bus dialog box is used for extending the data for a given bus record. You can specify additional generation, load, and shunt admittance. A typical use is the case where several owners have load at the same bus. Also, shunt specified on this record is considered to be fixed, rather than variable. See the + (plus) record [Continuation Bus Data \(+\)](#).

**Name.** An eight character maximum, alphanumeric string, plus a five character maximum real number, designating the name of the bus that this continuation data is associated with. **Code Type.** An option button that specifies the type of continuation record: +blank, +A, +C, +F +I, +N, +P, or +S. See the IPF Batch User's Guide for an explanation of these codes.

**Owner.** Three character maximum, alphanumeric string designating the owner of this particular load, shunt, etc. This will usually be different from the owner of the bus itself.

**Code Year.** Two character maximum, alphanumeric string. See [Continuation Bus Data \(+\)](#). for details.

**Load P.** Five character maximum, real number designating real load in megawatts (MW).

**Load Q.** Five character maximum, real number designating reactive load in megavoltamperes reactive (Mvar).

**Shunt P.** Four character maximum, real number designating the shunt admittance load in megawatts (MW) at the base kV of the bus.

Name

Owner  Zone  Type M

Number of Bridges

Smoothing Reactor  mh

Min Firing Angle  Degrees

Max Firing Angle  Degrees

Value Drop  Volts

Bridge Rating  Amps

Commutating Bus

Converter Type

Ignition Delay Angle  Degrees

Min Extinction Angle  Degrees

Converter DC Power  MW

Converter DC Voltage  kV

Add Modify Reset

Delete Outage Close

Fig. 2.10.17: Input Data Edit Box Showing BM Bus

Name

Code	Type	Owner	Code	Year
<input type="text" value="+"/>	<input type="text" value=""/>	<input type="text" value=""/>	<input type="text" value=""/>	<input type="text" value=""/>

<input type="text" value="Load P"/>	<input type="text" value="Load Q"/>
<input type="text" value="Shunt P"/>	<input type="text" value="Shunt Q"/>

<input type="text" value="Gen P"/>	<input type="text" value="Gen Qmax"/>	<input type="text" value="Gen Qmin"/>
------------------------------------	---------------------------------------	---------------------------------------

Fig. 2.10.18: Continuation Bus Dialog Box

**Shunt Q.** Four character maximum, real number designating the shunt reactance load in megavoltamperes reactive (MVar) at the base kV of the bus.

**Gen P.** Five character maximum, real number designating scheduled real power in megawatts (MW) as a real number.

**Gen Qmax.** Five character maximum, real number designating maximum reactive power in megawatts (MW).

**Gen Qmin.** Five character maximum, real number designating minimum reactive power in megawatts (MW).

### 2.10.21 Switched Reactance

The switched reactance bus dialog box is used for specifying steps in a switched reactance BX bus. See the X record *Switched Reactance (X)* for detailed information.

**Name.** An eight character maximum, alphanumeric string, plus a five character maximum real number, designating the name of the BX bus that this data is associated with.

**Remote Bus.** An eight character maximum, alphanumeric string, plus a five character maximum real number, designating the name of the remote bus to be voltage controlled.

**Owner.** A three character maximum, alphanumeric string designating the bus owner.

**Steps.** An integer from 1 to 9, designating the number of increments of shunt of this magnitude.

**MVAR.** A five character maximum, real number designating a block of switchable reactive shunt in megavoltamperes reactive (Mvar).

### 2.10.22 PQ Curve

The PQ Curve dialog box allows you to specify points for a generator reactive capability curve for a type BE, BG, BQ, BX, or BS bus. See the QP' record *Reactive Capability Curves (QP, QX, QN)* for detailed information.

To specify P Gen, Q Max, and Q Min values, type the values in the bottom text entry boxes. Click the Insert button to transfer the values from the text entry boxes to the list boxes above. Rows of values are associated across. Six rows of values are sufficient for most curves. Once you have values typed in and entered, you can replace or delete them, a row at a time.

**P Gen.** Five character maximum, real number designating a particular level of real power generation in megawatts (MW) which is to be associated with certain Q limits. Values may be specified in per unit on Pmax, or in MVA. All values for a curve must be specified the same way.

**Q Max.** Five character maximum, real number designating maximum reactive power (positive) in megavoltamperes reactive (Mvar) that can be produced by the generator when operating at this level of real power output. Values may be specified in per unit on Pmax, or in MVA. All values for a curve must be specified the same way.


**Q Min.** Five character maximum, real number designating minimum reactive power (negative) in megavoltamperes reactive (Mvar) that can be absorbed by the generator when operating at this level of real power output. Values may be specified in per unit on Pmax, or in MVA. All values for a curve must be specified the same way.

**Insert.** A button that inserts the values in the bottom text entry boxes into the text lists above. Replace. A button that replaces the selected row of values in the list above with the current values in the bottom text entry boxes.

**Delete.** A button that deletes the selected text list row of values.

**Active.** A radio button that makes the curve defined by the values in the text list rows active, that is, IPF uses the curve to determine what the Q limits will be, based on the current level of Pgen specified in the bus record.

**Inactive.** A radio button that makes the curve inactive, that is, IPF does not calculate new Q limits whenever Pgen is changed, but uses whatever it currently has stored.

Name **G4 HIGH** **230.** **Bus** 

Remote Bus   Owner

	Steps	MVAR
Increment 1	<input type="checkbox"/>	<input type="text"/>
Increment 2	<input type="checkbox"/>	<input type="text"/>
Increment 3	<input type="checkbox"/>	<input type="text"/>
Increment 4	<input type="checkbox"/>	<input type="text"/>
Increment 5	<input type="checkbox"/>	<input type="text"/>
Increment 6	<input type="checkbox"/>	<input type="text"/>
Increment 7	<input type="checkbox"/>	<input type="text"/>
Increment 8	<input type="checkbox"/>	<input type="text"/>

**Add** **Modify** **Reset**

**Delete** **Outage** **Close**

Fig. 2.10.19: Switched Reactance Dialog Box

INPUT WITH EDIT BOX

Name:

P Gen	Q Max	Q Min
140.00	130.00	-100.0
165.00	120.00	-80.00
180.00	5.0000	-6.000
<input type="text"/>	<input type="text"/>	<input type="text"/>

☒ Active    ☐ Inactive  
☒ MVA    ☐ Per Unit

Fig. 2.10.20: P-Q Generation Dialog Box

**MVA.** The values for the PQ curves may be specified in MVA or per unit. Clicking the MVA radio button tells the program to expect values in MVA.

**Per Unit.** The values for the PQ curves may be specified in MVA or per unit. Clicking the Per Unit radio button tells the program to expect values in per unit on Pmax.

**Add.** A button that adds a new three-record point set to the current curve data for this bus.

**Modify.** A button that modifies the curve data. (Not available.)

**Reset.** A button that restores text box values to their original values (before any changes were made).

**Delete.** A button that deletes (removes) the curve data from the database.

**Outage.** (Not applicable.)

**Close.** A button that causes the dialog box to close and disappear from the display without making any modifications.

### 2.10.23 Sectionalization

The Sectionalize Bus dialog box allows you to split a bus to create two buses, with existing branches divided between them. You can sectionalize a bus at any time. You get to this dialog box from the Sectionalize button in the Input Data Edit dialog box for the bus you want to split.

When the Sectionalize Bus dialog box first comes up, it assumes the current bus name and information from the Input Data Edit dialog box. Note that the name of the current bus appears in both text boxes found at the top of the dialog box. You change the name in the right-hand box to create a new bus record, which will inherit the bus type and voltage of the old bus.

Once you have changed the bus name to a new one, you can arrange the branch information in the list boxes to define the new connections. The list box under the left-hand text box applies to the bus name on the left side, and the list on the right to the right-hand bus name. If you click on a record in either box, it will be transferred to the other. Use the horizontal and vertical scroll bars to see information that is hidden.

After the two buses and their associated branches are satisfactory, you can optionally press the Bus Tie button to create a “bus tie” record, which is a line with impedance  $0.0 + j0.00001$  between the new bus and the old one. The bus tie record will appear in the text box.

**Sectionalize Bus 1.** This text box contains the bus type, name, and base kV of the current bus you are working with in the Input Data Edit Box.

**Sectionalize Bus 2.** This text box initially contains the name of the current bus. You can change this to be any new bus name. The new bus inherits the base kV, ownership, and bus type of Bus 1.

**Bus Tie.** Click this button if you want to tie the two buses with a low impedance tie line. This creates a bus tie record. The branch record shows up in the text box. OK. Click this button to cause the new data to be saved in the memory-resident bus and branch database. No changing action occurs until you click OK. The dialog box closes and returns you to the Input Data Edit Box.

**Reset.** Click this button if you want to return to the initial state of a just opened dialog box. All changes that you have made are erased and returned to initial conditions.

**Close.** Click this button if you have decided that no save action is necessary, that is, you do not want to make any sectionalization changes to the memory-resident bus and branch database. Clicking this button closes the dialog box and returns you to the Input Data Edit Box.



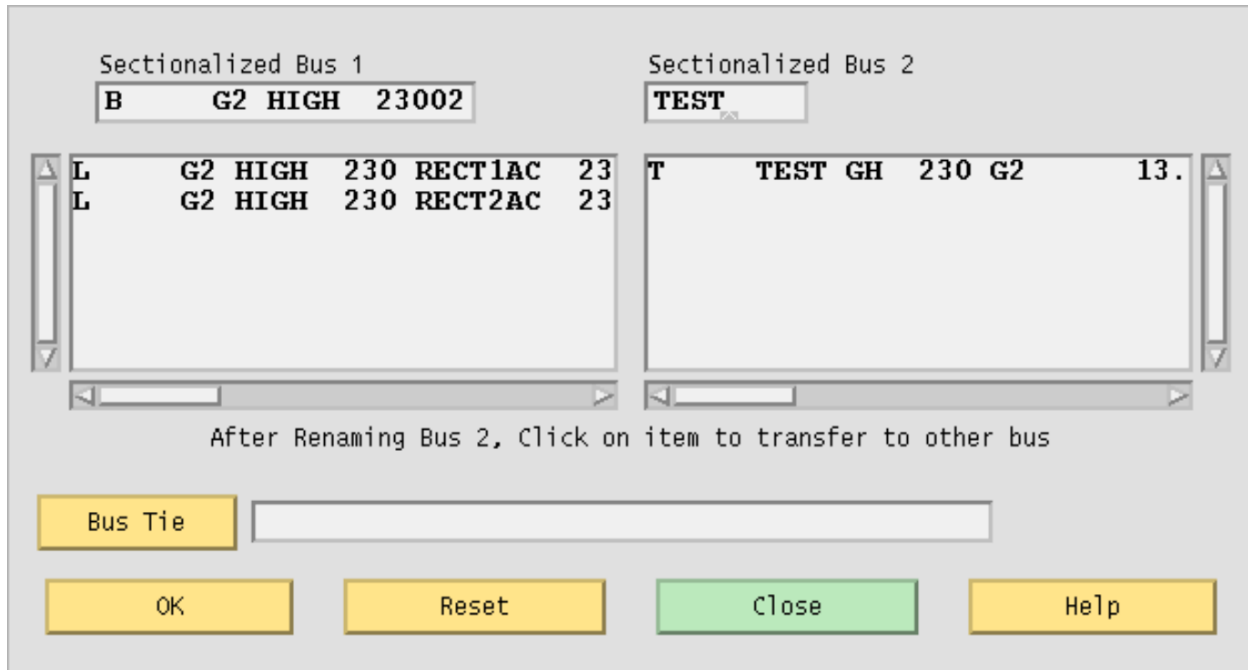


Fig. 2.10.21: Sectionalize Bus Dialog Box

### 2.10.24 Line Tapping (may not be available)

Tapping a line means to service a new load by creating a new tap point bus on an existing line. The tapped line is effectively segmented into two lines, separated with a newly created bus. If the load is remote from the tapped point, an additional line and bus will be necessary. The new load and the new bus are connected by a new line. As with many line operations, you access line tapping through a bus that the line is connected to.

**Bus 1 Line Data.** This text box displays the lines between bus 1 and the tapped bus. Initially, it contains all the lines between Bus 1 and Bus 2.

**Bus 2 Line Data.** This text box displays the lines between bus 1 and the tapped bus. Initially, it is empty.

**Scale Value Radio Buttons.** These ratio buttons identify the line tap point in terms of three different criteria: (1) percent of reactance from bus 1, (2) the distance from bus 1, (3) the section number. Note that transformers and series capacitors ( $X < 0$ ) have zero (0) length and cannot be tapped.

**Scale Slide.** A dynamic, moveable slider that shows the proportion of the selected scale on the line between Bus 1 and Bus 2. The slider value changes according to what scale value radio button is currently active.

**Base 1 Name.** A bus name and base kV bus identifier specifying the terminal 1 bus of the line to be tapped.

**Base 2 Name.** A bus name and base kV bus identifier specifying the terminal 2 bus of the line to be tapped.

**Reverse Scale.** A button that flips the scale values from one end to the other.

**Tapped Bus Name.** The name and base kV of the new bus created at the tap point.

**Send.** Click this button to cause the line tapping operation data to be saved in the memory-resident bus and branch database. The dialog box closes and returns you to the Input Data Edit Box.

**Calculate.** Click this button to see the effects of the slider operation. This action does not send any data to Powerflow.

**Cancel.** A button that closes the dialog box and causes no further action.



The dialog box is titled "LINE TAPPING". It contains two main sections: "Bus 1 Line Data" and "Bus 2 Line Data", each with a large empty text area. Below these is a row of four radio buttons: "Percentage" (selected), "Miles", "Kilometers", and "Section". A horizontal slider is positioned below the radio buttons, with a value of "50.0" displayed in the center. Below the slider, there are three text input fields: "ADELAIDE 345" on the left, "DORAH 345" on the right, and "HEMBUS 345." in the center. Below the left and right fields are the labels "Bus 1 Name" and "Bus 2 Name" respectively. Below the center field is the label "Tapped Bus Name". At the bottom of the dialog box, there are five buttons: "Send", "Calculate", "Cancel", "Close", and "Help".

Field	Value
Bus 1 Line Data	
Bus 2 Line Data	
Percentage	<input checked="" type="radio"/>
Miles	<input type="radio"/>
Kilometers	<input type="radio"/>
Section	<input type="radio"/>
Slider Value	50.0
Bus 1 Name	ADELAIDE 345
Tapped Bus Name	HEMBUS 345.
Bus 2 Name	DORAH 345

Fig. 2.10.22: Line Tapping Dialog Box

**Close.** Click this button if you have decided that no save action is necessary, that is, you do not want to make any sectionalization changes to the memory-resident bus and branch database. Clicking this button closes the dialog box and returns you to the Input Data Edit Box.

**Help.** (Not yet implemented)

### 2.10.25 Transmission Line

The transmission line dialog box specifies the identification and electrical characteristics of a balanced pi line, section of a line, or series capacitor. See the L record *Balanced Transmission Line Branch (L)*.

**Name.** Two eight character maximum, alphanumeric strings designating the buses. The strings must start with an alpha character. The first name is placed in the first text box from the left. The next text box should have a four character maximum numeric string representing the bus's base kV rating. The second bus name and its base kV are to the right of the first.

**Metering.** An integer (or blank) flag having three possible values: 1 means to meter at the bus 1 end; 2 means to meter at the bus 2 end; and blank means to let the program decide on the following criteria — (1) when bus ownership differs from bus ownership, meter at the point where line ownership differs from bus ownership, or (2) when both buses have the same ownership, meter at bus 1 location.

**Owner.** A three character alphanumeric code representing ownership of the branch. **Circuit ID.** A single alphanumeric character representing the circuit identification. **Section.** An integer (1-9) representing the section number for making an equivalent for series elements. The elements are assembled in ascending numeric order. This may be blank or zero if the line has only one section.

**Resistance (R).** A six digit real number representing the per unit resistance R.

**Reactance (X).** A six digit real number representing the per unit reactance X.

**Admittance (G/2).** A six digit real number representing the per unit admittance G.

**Susceptance (B/2).** A six digit real number representing the per unit susceptance B.

**Number of Parallels.** An integer representing the number of parallel circuits represented by this record.

**Miles.** A real number indicating the line length. Note: if a branch is composed of individual sections, then the total line length is the sum of mileage of each section. Also, note that series capacitors ( $X < 0$ ) have no mileage.

**Current Ratings.** Real numbers that are conductor current ratings in amps. Nominal is the normal rating based on the line construction and conductor size. Thermal takes into account the effect of ambient temperature and other environmental factors upon the maximum permissible temperature of the conductor, usually for short time periods. Bottleneck is the minimum rating of the line including other series connected components, such as circuit breakers, fuses, or disconnect switches.

**Calculate Impedance.** A button that allows you to compute the electrical parameters given the conductor size and type, and tower geometry and length.

**Tap Line.** A button that allows you to tap a line with a newly added bus.

**Add.** A button that adds a new record to the database.

**Modify.** A button that modifies the record.

**Reset.** A button that restores text box values to their original values (before any changes were made).

**Delete.** A button that deletes (removes) a record from the database.


**Outage.** (Not yet implemented.)

**Close.** A button that causes the dialog box to close and disappear from the display without making any modifications to the record.

INPUT WITH EDIT BOX

Name: FOURCORN 345. L SHAPROCK 345 1

Notering: 1 Owner: MAP

Circuit ID: 1 Section: 

PER UNIT

$Z_{pi} = R + jX$   $Y_{pi} = G + jB/2$

R: 0.00028 G/2:

X: 0.00417 B/2: 0.03214

Number of Parallels: ☐ Miles:

CURRENT RATINGS

Nominal:  Thermal:  Bottleneck:

Calculate Impedance Tap Line

Add Modify Reset

Delete Outage Close

Fig. 2.10.23: Transmission Line Dialog Box

## 2.10.26 Phase Shifter

The Phase Shifter dialog box allows you to add data for phase shifting transformers. See the T *Transformer Data (T, TP)* and R records *Regulating Transformer (R, RV, RQ, RP, RN, RM)* for more details.

**Name.** Two eight character maximum, alphanumeric strings designating the buses. The strings must start with an alpha character. The first name is placed in the first text box from the left. The next text box should have a four character maximum numeric string representing the bus's base kV rating. The second bus name and its base kV are to the right of the first.

**Metering.** An integer (or blank) flag having three possible values: 1 means to meter at the bus 1 end; 2 means to meter at the bus 2 end; and blank means to let the program decide on the following criteria — (1) when bus ownership differs from bus ownership, meter at the point where line ownership differs from bus ownership, or (2) when both buses have the same ownership, meter at bus 1 location.

**Section.** An integer (1-9) representing the section number for making an equivalent for series elements. The elements are assembled in ascending numeric order.

**Circuit ID.** A single alphanumeric character representing the circuit identification.

**Owner.** A three character alphanumeric code representing ownership.

**Parallels.** An integer representing the number of parallel transformers in this record.

**Resistance (R).** A six digit real number representing per unit equivalent resistance R due to copper loss.

**Reactance (X).** A six digit real number representing per unit leakage reactance X.

**Admittance (G).** A six digit real number representing per unit shunt equivalent core loss conductance G (iron losses).

**Susceptance (B).** A six digit real number representing per unit shunt magnetizing susceptance B. This is always converted to a negative number by the program.

**Phase Shift.** A five digit real number representing the fixed phase shift in degrees that describes bus 1 relative to bus 2.

**Tap 2 kV.** A five digit real number representing the fixed bus 2 tap. It is possible for a transformer to have both a phase shift and a tap.

**MVA Ratings.** All MVA ratings (Nominal, Thermal, Bottleneck, and Emergency) are represented by four digit real numbers. Nominal is the normal rating based on the construction. Thermal takes into account the effect of ambient temperature and other environmental factors upon the maximum permissible temperature of the conductor, usually for short time periods. Bottleneck is the minimum rating of the transformer including other series connected components, such as circuit breakers, fuses, or disconnect switches.

**Add.** A button that adds a new record to the database.

**Modify.** A button that modifies the record.

**Reset.** A button that restores text box values to their original values.

**Delete.** A button that deletes (removes) a record from the database.

**Outage.** (Not yet implemented.)

**Close.** A button that causes the dialog box to close and disappear from the display without making any modifications to the record.

INPUT WITH EDIT BOX

Name   TP BLOS PHA 381 1

Metering ☐ Section ☐

Circuit ID  Diner  Parallels

PER UNIT

$Z_{pi} = R + jX$

R

X

$Y_{pi} = G + jB$

G

B

Phase Shift

Tap 2 kV

IPA RATINGS

Normal

Thermal

Bottleneck

Emergency

ADD Modify Reset

Delete Change Close

Fig. 2.10.24: Phase Shifter Dialog Box

## 2.10.27 Transformer

The transformer dialog box allows you to add data for fixed tap transformers. See the T record *Transformer Data (T, TP)*.

**Name.** Two eight character maximum, alphanumeric strings designating connected buses. The strings must start with an alpha character. The first name is placed in the first text box from the left. The next text box should have a four character maximum numeric string representing the bus's base kV rating. The second bus name and its base kV are to the right of the first.

**Metering.** An integer (or blank) flag having three possible values: 1 means to meter at the bus 1 end; 2 means to meter at the bus 2 end; and blank means to let the program decide on the following criteria — (1) when bus ownership differs from bus ownership, meter at the point where line ownership differs from bus ownership, or (2) when both buses have the same ownership, meter at bus 1 location.

**Section.** An integer (1-9) representing the section number for making an equivalent for series elements. This may be zero or blank if the branch has only one section.

**Circuit ID.** An alphanumeric character representing the circuit identification.

**Owner.** A three character alphanumeric code representing ownership.

**Parallels.** An integer representing the number of parallel transformer banks in this record.

**Resistance (R).** A six digit real number representing per unit equivalent resistance R due to copper loss.

**Reactance (X).** A six digit real number representing per unit leakage reactance X.

**Admittance (G).** A six digit real number representing per unit shunt equivalent core loss conductance G (iron losses).

**Susceptance (B).** A six digit real number representing per unit shunt magnetizing susceptance B. This is always converted to a negative number by the program.

**Tap 1 kV.** A five digit real number representing the fixed bus 1 tap.

**Tap 2 kV.** A five digit real number representing the fixed bus 2 tap.

**MVA Ratings.** All MVA ratings (Nominal, Thermal, Bottleneck, and Emergency) are represented by four digit real numbers. Nominal is the normal rating based on the construction. Thermal takes into account the effect of ambient temperature and other environmental factors upon the maximum permissible temperature of the conductor, usually for short time periods. Bottleneck is the minimum rating of the transformer including other series connected components, such as circuit breakers, fuses, or disconnect switches.

**Add.** A button that adds a new record to the database.

**Modify.** A button that modifies the record.

**Reset.** A button that restores text box values to their original values (before any changes were made).

**Delete.** A button that deletes (removes) a record from the database.

**Outage.** (Not yet implemented.)

**Close.** A button that causes the dialog box to close and disappear from the display without making any modifications to the record.

INPUT WITH EDIT BOX

Name: FOURCORN 345. T: FONGWACC22.0 1

Metering ☐ Section ☐

Circuit ID: 1 Owner: Parallels: ☐

PER UNIT

$Z_{pi} = R + jX$   $Y_{pi} = G + jB$

R: 0.00032 G: 0.00587

X: 0.01611

Tap 1 kV: 353.63 Tap 2 kV: 22.00

IFIA RATINGS

Nominal: 924 Thermal:

Bottleneck:  Emergency:

Add Modify Reset

Delete Outage Close

Fig. 2.10.25: Transformer Dialog box



## 2.10.28 Regulating Transformer

The regulating transformer dialog box allows you to add data for regulating transformers. See the R record *Regulating Transformer (R, RV, RQ, RP, RN, RM)*.

**Name.** Two eight character maximum, alphanumeric strings designating connected buses. The strings must start with an alpha character. The first name is placed in the first text box from the left. The next text box should have a four character maximum numeric string representing the bus's base kV rating. The second bus name and its base kV are to the right of the first.

**Owner.** A three character alphanumeric code representing ownership.

**R (subtypes).** An option button allowing you to choose types R-blank, RV, RQ,RP, RN, or RM. See the R record in the IPF Batch User's Guide for a description of these types.

**Low Alpha Fixed.** A radio button that identifies the fixed tap side as at the low alpha order bus name terminal. Note that this field is necessary only to resolve ambiguity if Min Tap and Max Tap cannot establish the variable tap side. Low Alpha Fixed is the default.

**Bus 1 Variable.** A radio button that identifies the variable tap side as at the bus 1 terminal. Note that this field is necessary only to resolve ambiguity if Min Tap and Max Tap cannot establish the variable tap side.

**Bus 2 Variable.** A radio button that identifies the variable tap side as at the bus 2 terminal. Note that this field is necessary only to resolve ambiguity if Min Tap and Max Tap cannot establish the variable tap side.

**Remote Bus.** An eight character maximum, alphanumeric string designating the remote bus to be voltage-controlled.

**Min Tap.** A real number in kV that specifies the minimum tap on the variable tap side.

**Max Tap.** A real number in kV that specifies the maximum tap on the variable tap side.

**Number of Taps.** An integer specifying the number of taps on the variable tap side. This must be greater than one for discrete taps; zero (0) indicates continuous taps. Zero is the default.

**Add.** A button that adds a new record to the database.

**Modify.** A button that modifies the record.

**Reset.** A button that restores text box values to their original values (before any changes were made).

**Delete.** A button that deletes (removes) a record from the database.

**Outage.** (Not yet implemented.)

**Close.** A button that causes the dialog box to close and disappear from the display without making any modifications to the record.

## 2.10.29 Equivalent Network

The equivalent network dialog box allows you to add data for an equivalent, unbalanced pi transmission line branch. See the E record *Equivalent Transmission Line Branch (E)*.

**Name.** Two eight character maximum, alphanumeric strings designating connecting buses. The strings must start with an alpha character. The first name is placed in the first text box from the left. The next text box should have a four character maximum numeric string representing the bus's base kV rating. The second bus name and its base kV are to the right of the first.

**Metering.** An integer (or blank) flag having three possible values: 1 means to meter at the bus 1 end; 2 means to meter at the bus 2 end; and blank means to let the program decide on the following criteria — (1) when bus ownership differs from bus ownership, meter at the point where line ownership differs from bus ownership, or (2) when both buses have the same ownership, meter at bus 1 location.

INPUT WITH EDIT BOX

Name:

Owner:

☐ Low Alpha Fixed  
☐ Bus 1 Variable  
☐ Bus 2 Variable

Remote Bus:

Max Phase Shift:  Min Phase Shift:

Number of Shifts:

Min Flow:  Max Flow:

Fig. 2.10.26: Regulating Transformer Dialog Box

INPUT DATA EDIT BOX

Name: FOURCORN 345. E CHOLLA 345 1 1

Notering ☐ Owner ☐

Circuit ID 1 Section 1

PER UNIT

$Z_{pi} = R + jX$

R 0.00836

X 0.08117

$Y_{pi1} = G1 + jB1$   $Y_{pi2} = G2 + jB2$

G1 0.00060 G2 0.00060

B1 0.64422 B2 0.64422

Number of Parallels ☐

CURRENT RATINGS

Nominal Thermal Bottleneck

Add Modify Reset

Delete Outage Close

Fig. 2.10.27: Equivalent Network Dialog Box

**Owner.** A three character alphanumeric code representing ownership.

**Circuit ID.** An alphanumeric character representing the circuit identification.

**Section.** An integer (1-9) representing the section number for making an equivalent for series elements. The elements are assembled in ascending numeric order. This may be blank or zero if the line has only one section.

**Resistance (R).** A six digit real number representing the per unit resistance R through the branch from bus 1 to bus2.

**Reactance (X).** A six digit real number representing the per unit reactance X through the branch from bus 1 to bus 2.

**Admittance (G1).** A six digit real number representing the line's per unit shunt conductance G at the bus 1 terminal.

**Susceptance (B1).** A six digit real number representing the line's per unit shunt susceptance B at the bus 1 terminal.

**Admittance (G2).** A six digit real number representing the line's per unit shunt conductance G at the bus 2 terminal.

**Susceptance (B2).** A six digit real number representing the line's per unit shunt susceptance B at the bus 2 terminal.

**Number of Parallels.** An integer representing the number of parallel transformer banks in this record.

**Current Ratings.** Real numbers that are conductor current ratings in amps. Nominal is the normal rating based on the line construction and conductor size. Thermal takes into account the ambient temperature and other environmental factors upon the maximum permissible temperature of the conductor, usually for short time periods. Bottleneck is the minimum rating of the line including other series connected components, such as circuit breakers, fuses, or disconnect switches.

**Add.** A button that adds a new record to the database.

**Modify.** A button that modifies the record.

**Reset.** A button that restores text box values to their original values (before any changes were made).

**Delete.** A button that deletes (removes) a record from the database.

**Outage.** (Not yet implemented.)

**Close.** A button that causes the dialog box to close and disappear from the display without making any modifications to the record.

### 2.10.30 Menu Commands

The commands described in this section are all accessible from the menu bar in the main window. The commands are arranged alphabetically. Each command entry is found at the top of a page and shows you which menu it is on by including the menu name in parentheses. For example, the entry Alpha Search (View) means that the Alpha Search command is found on the View menu.

See the table below for a quick look at the main window menu commands.

Page	Command	Description
4-50	ALPHA SEARCH (View)	Finds a specific bus by name.
4-51	AREA/INTERCHANGE (Edit)	Allows editing of area/interchange records.
4-54	AUTO CFLOW (Process)	Allows execution of CFLOW programs by users.
4-57	BENDING POINTS (View)	Turns on (and off) display of the capital B denoting a bending point.
4-58	COLOR SCHEME (View)	Switches between line display by overload or by nominal kV.
4-59	COMMAND DIALOG (View)	Allows typing of PCL commands for PF.
4-61	ERROR MESSAGES (Help)	Displays IPF error messages.
4-62	EXIT (File)	Exits IPF. Same as Exit button.
4-63	GENERAL (Help)	Displays on-line help text.
4-65	NETWORK DATA EDIT (Edit)	Allows editing of network data.
4-70	OPEN (File)	Allows loading of IPF files.
4-74	PF ID/DESCRIPTION (Edit)	Allows creating and saving of user case description.
4-76	PLOT OPTIONS (File)	Allows changing of printer and diagram attributes and the printer device itself.
4-83	PRINT PLOT (File)	Prints a case diagram to the currently set printer
4-84	REPORTS (View)	Creates some standard PF reports and allows display viewing, hard copy printing, and writing to files.
4-91	RUN CFLOW (Process)	Allows the running of CFLOW C programs from IPF. (Intended for CFLOW programmers.)
4-92	SAVE (File)	Saves change, base case, and coordinate files.
4-96	SOLUTION DATA OFF (View)	Rewrites the display without rewriting the current solution data.
4-97	SOLUTION DATA ON (View)	Rewrites the display with the current solution data.
4-98	SOLVE CASE (Process)	Initiates the solution of a resident base case

### Alpha Search (View)

The Alpha Search dialog box allows you to find any bus within the currently loaded base case data. Any bus selected when you close this dialog box becomes the currently selected bus until some other selection process changes it.

When you open the dialog box from the View menu, you see a text box below Search Bus Name kV. Type in this box any character or string of characters matching the first part of the name of the bus you are looking for. Thus, if you are looking for CASCADTP, you type in C, CA, or CAS to take you to the part of the bus list starting with C, CA, or CAS. Note that the search function is case sensitive; that is, c and C are not the same. The search function immediately begins searching as soon as you enter a character in the text box.

You can also use the scroll bar to go up or down the list to visually identify the bus you are looking for, and then select the desired bus by clicking it

**Search Bus Name kV.** Use this text box to type a string of letters at the beginning of the bus name you are looking for.

**Close.** Clicking this button makes the last selected bus name into the currently selected bus. The dialog box then closes.

### Area/Interchange (Edit)

For area or intertie studies, you can add, modify, or delete areas or interties. You do this through the main window Area/Interchange command.

The Area/Interchange dialog box includes all the area and intertie records from the currently resident base case - one record per line in a list box. Selecting a line puts the line in the Selection text box. Then press the Edit Area/Intertie Record button to bring up a dialog box with the data in it.

**List of Area Control and Intertie records.** This scrolling list contains all the area and intertie records from the currently resident base case. Area records are listed first followed by intertie records. Scroll through the list to find the one you want. Select it by clicking on it.

**Selection.** This text box contains the currently selected record from the list of records above. This text will not change when you edit a record, so you can compare the new record with the old.

**Edit Area/Intertie Record.** Clicking this button opens a dialog box where you can edit the currently selected area or intertie record (line).

**Apply.** Not available.

**Close.** Clicking this button causes the Area/Interchange dialog box to close and disappear from the display.

**Create New.** Clicking this button brings up the a blank dialog box so you can create a new area or intertie record.

**Interchange Area.** A ten character maximum name designating an interchange area of a network.

**\*\*Area Slack Bus.** An eight character maximum name designating the area slack bus plus a four character maximum real number representing the base kV rating of the slack bus.

**Scheduled Export.** An eight character maximum real number designating the scheduled export in MW. Negative denotes inflow.

**Zone.** A two character alphanumeric designating zone. The zone at the extreme left must be filled in; others may be blank or filled.

**Max PU Volt.** A four character maximum real number designating maximum per unit voltage for this area.

**Min PU Volt.** A four character maximum real number designating minimum per unit voltage for this area.

**Modify.** A button that modifies and updates in place a record (line) in the Area/Interchange dialog box. The data in the selection box is not changed.

**Add.** A button that adds a new record (line) to the list in the Area/Interchange dialog box.



Fig. 2.10.28: Alpha Search Dialog box

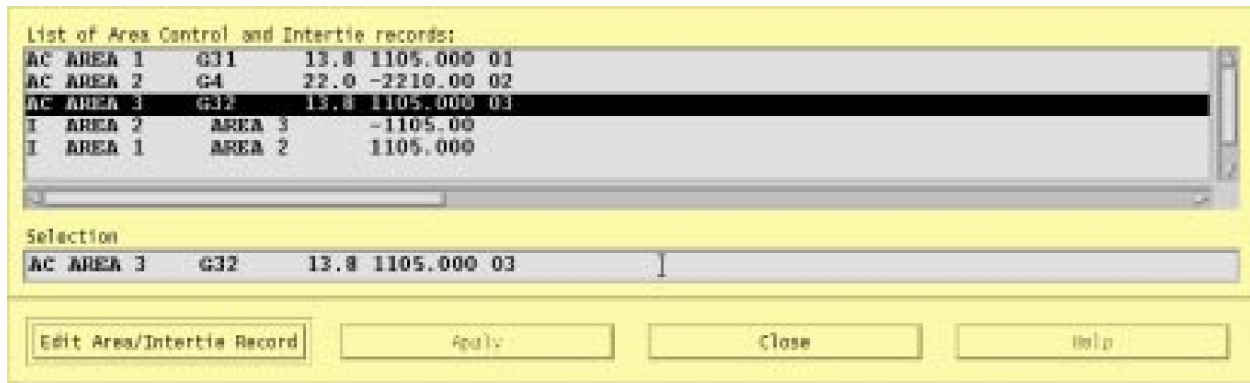


Fig. 2.10.29: Area/Interchange Edit Dialog Box

Fig. 2.10.30: Area/Interchange Record Dialog Box

**Delete.** A button that deletes (removes) a record (line) from the list in the Area/Interchange dialog box.

**Reset.** A button that restores text box values to their original values (before any changes were made).

**Close.** Clicking this button causes the Area/Interchange dialog box to close and disappear from the display without making any changes to the Area/Interchange records.

**Area Name 1.** A ten character name designating an area of a network.

**Area Name 2.** A ten character name designating an area of a network.

**Sched Export Number for Sched Interchange.** An eight character maximum real number representing scheduled power transfer from Area Name 1 to Area Name 2.

**Modify.** A button that modifies and updates in place a record (line) in the Area/Intertie Selection dialog box.

**Add.** A button that adds a new record (line) to the list in the Area/Intertie Selection dialog box.

**Delete.** A button that deletes (removes) a record (line) from the list in the Area/Intertie Selection dialog box.

**Reset.** A button that returns text box values to their original values (before any changes were made).

**Close.** Clicking this button causes the Intertie Record dialog box to close without making any changes to the Area/Interchange dialog box records.

### Auto CFLOW (Process)

CFLOW files are C programs using the IPF CFLOW library of routines. These routines enable you to access the base case memory-resident data in IPF. Many CFLOW programs are designed to collect information for specialized reports that are not built into IPF in the Reports command. See *CFLOW C API (libcflow)* for more information.

There are two ways to execute a CFLOW program: Auto CFLOW is one; Run CFLOW is the other. Refer also to Run CFLOW later in this section so that you choose the appropriate CFLOW execution command.

The Auto CFLOW command is used primarily by CFLOW program users (rather than programmers) for CFLOW programs that are fully debugged and “production-grade.” These CFLOW programs create windows for I/O themselves or are embedded in script or command file utilities that do. Thus, any I/O to or from the screen is program I/O only and not potentially confusing for a user. Alternatively, the CFLOW program may have no screen I/O and simply produce file output for a report. See Figure 4-21.



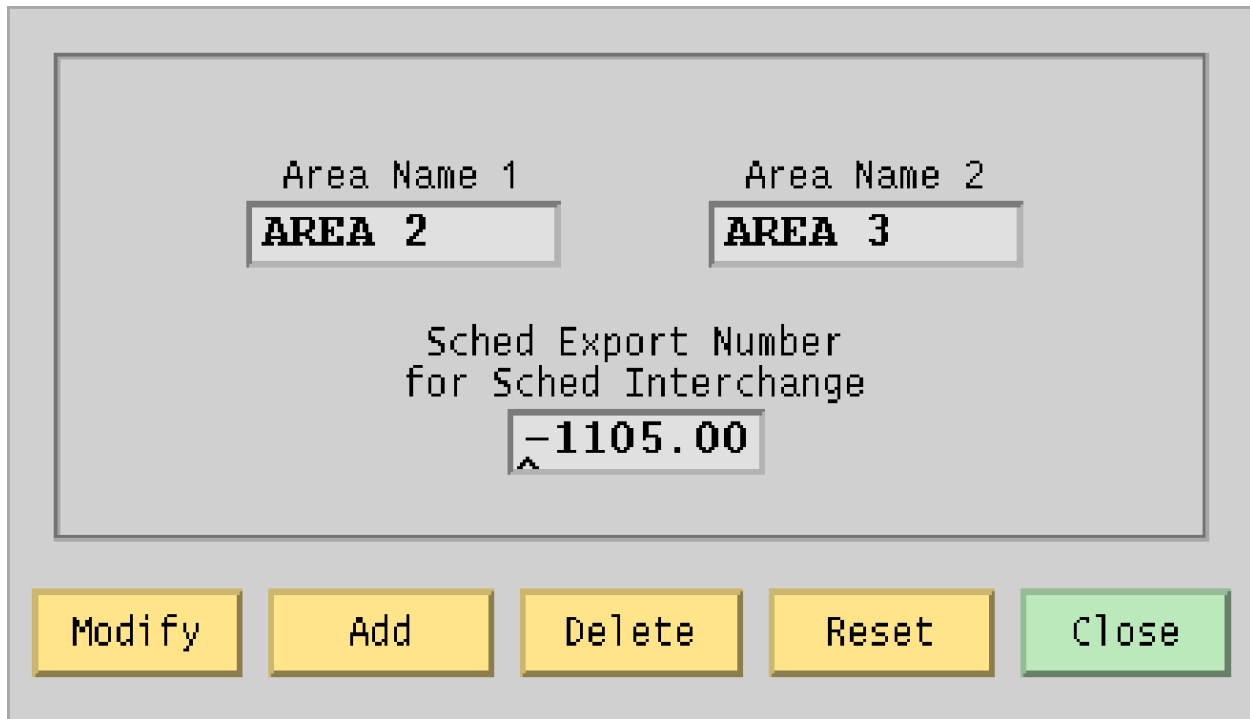


Fig. 2.10.31: Intertie Record Dialog Box

Programs that read/write to `stdin` or `stdout` would have their I/O intermixed in the same terminal window as the gui and `ipfsrv` server. Such programs should be run with the Run CFLOW command not with the Auto CFLOW command.

**To use the Auto CFLOW command:**

1. Start up IPF and load a base case file.
2. Select the Auto CFLOW command and (if necessary) specify a socket id number (any integer between 1024 and 4096). Generally the default socket number will be OK.
3. Use the file selection dialog box to find and select a CFLOW program. Double click the program name or click Launch CFLOW to run the program.

**Note:** When you have launched a CFLOW program, you cannot do anything else in the GUI until the program is finished running. Also if the CFLOW program fails, control is returned to IPF. However, if the CFLOW program hangs (as in an infinite loop), you need to kill the CFLOW process through operating system resources. (For example, in Unix, this can be done with the `kill` command.) See your computer system documentation or your system administrator for help.

**Socket Number.** This is the number of the TCP/IP socket which CFLOW will use to connect to `ipfsrv`. Normally, you should not have to change it.

**CLFLOW Program Arguments.** If the particular program you want to run requires input arguments, enter them here. You cannot, however, use this to redirect input or output.

**Filter.** The file name text box contains a “filter” that selects categories of file names. You can directly modify the file name text by clicking in the Filter box and then typing in new text. Then click the Filter button below to apply the new filter.

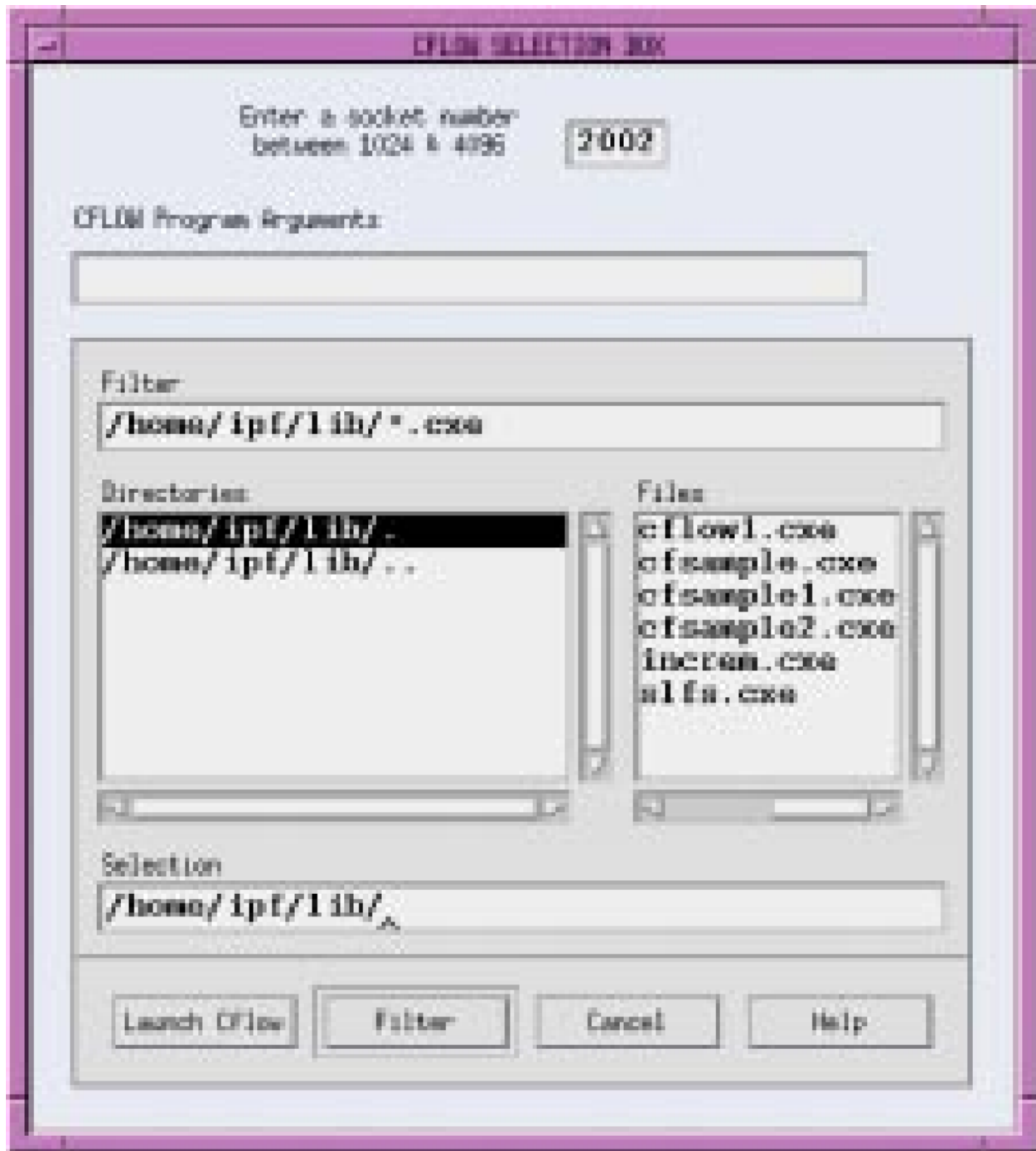


Fig. 2.10.32: Auto CFLOW File Dialog box

**Directories.** This list component contains directory names. You cannot modify these directory names by selecting them and typing. Use the scroll bars at the side and bottom to move the list up and down or back and forth. The UNIX operating system arranges directories in a “hierarchical” way. You move up this hierarchical tree structure by double clicking the directory name ending with a period-period (..). You move down by selecting the name of the directory you want to move into.

**Files.** This list component contains file names that are within the directory named at the left that ends with a period (.) and that satisfy the filter criterion specified above in Filter. You cannot modify these file names by selecting them and typing. Use the scroll bars at the side and bottom to move the file names up and down or back and forth.

You select a file name by clicking once on the file name. This puts the selected file name in the Selection file name text box below.

**Selection.** This file name text box contains the file name selected by clicking a file name in the Files file list. Or, since it is a text box, you can directly modify the file name by selecting text and typing replacement text. Be sure to type an exact file name and not a wildcard character such as the asterisk (\*) as part of the file name.

**Launch C Flow.** This button causes whatever file name is in the Selection text box to be passed to the operating system as an executable file. The file is then run.

**Filter.** Clicking this button causes all file names satisfying the filter file name text in Filter to appear in the Files file list. It also puts the currently selected directory name without the file name in the Selection text box. You must select a file name from the Files file list to select a specific file name.

**Cancel.** Clicking this button causes the Auto CFLOW File dialog box to close and disappear from the display. No directories or file names are changed with a Cancel action. Thus, if you immediately open the dialog box after a Cancel action, you return to the state you just left. The directories and files names are not returned to some default state.

**Help.** (Not yet implemented).

### Bending Points (View)

This command toggles on or off the display of a capital B at line bending points. The display of the capital B is simply to make line bending points completely apparent at a quick glance.

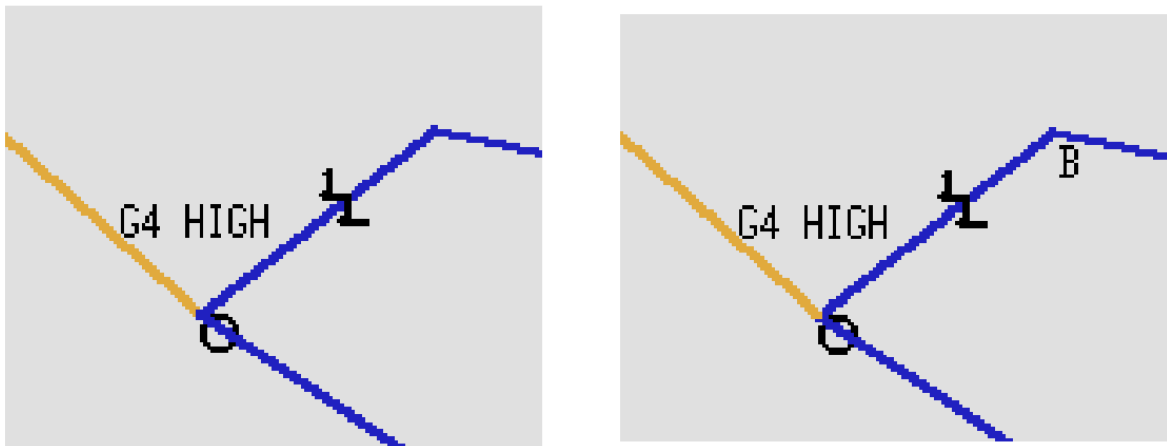


Fig. 2.10.33: Bending Points On and Off

### Color Scheme (View)

This command causes the display of lines to switch between two modes: Color by kV or Color by Overload. Color by kV causes the display of lines to match the ranges that are keyed in the lower left-hand corner Branch Color Key. Color by Overload causes the display of lines to match the ranges that are keyed in the Overloaded Branch Key. These two keys are found in the same location in the main window and change when you change the command.

In the Color by Overload mode, you can type in percentages of overload. The figure below shows 90% for Mild Overload, 100% for Moderate Overload, and 110% for Extreme Overload.

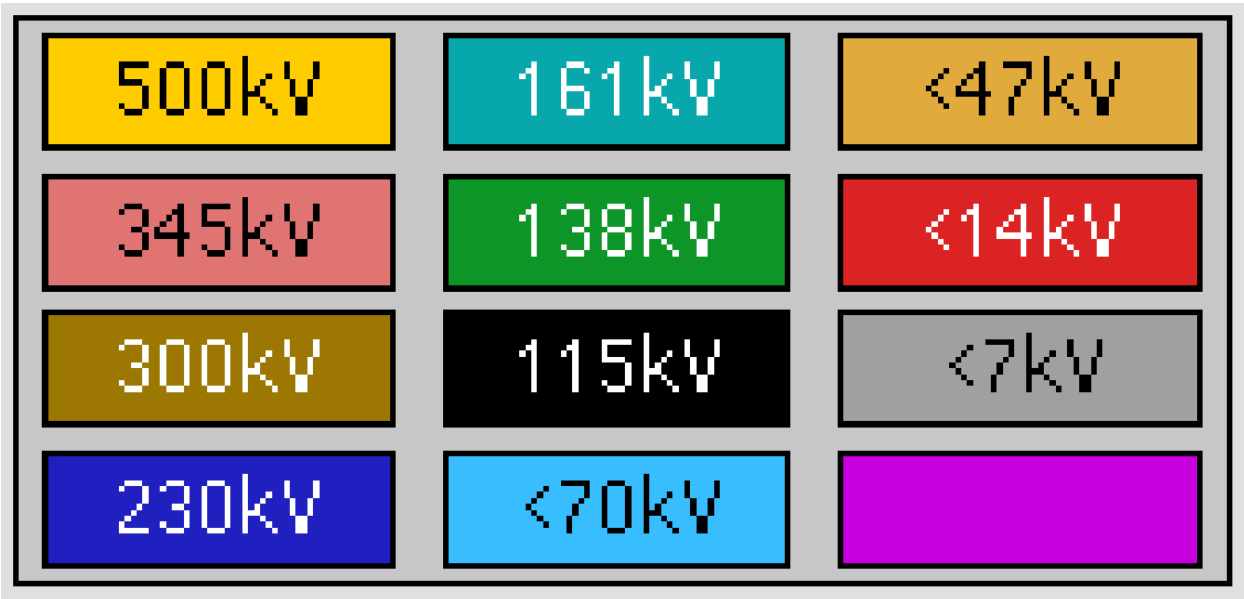


Fig. 2.10.34: Branch Color Key

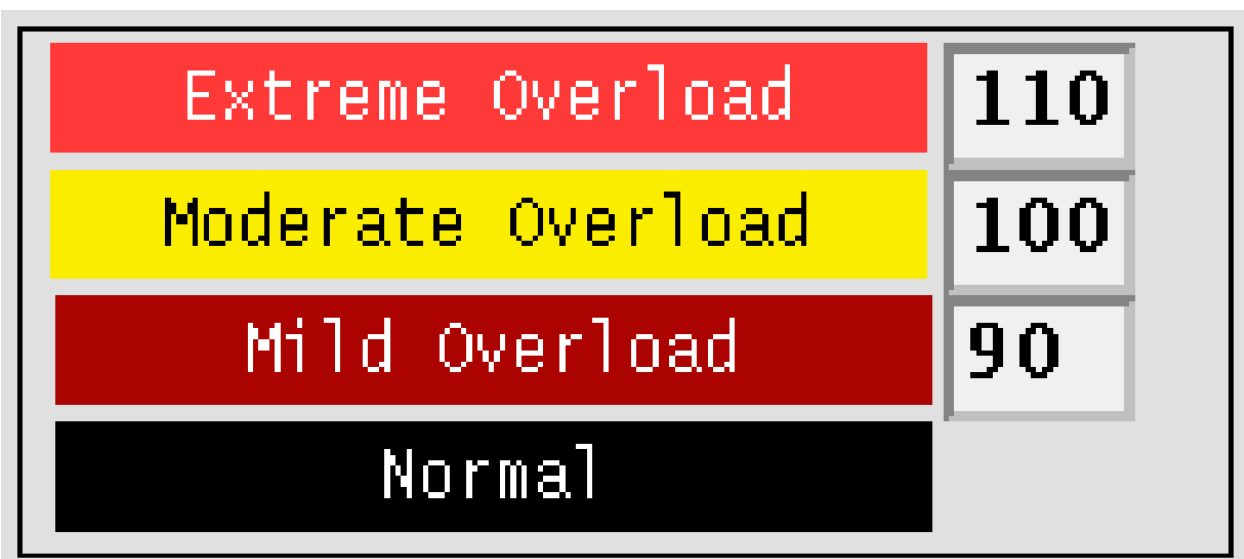


Fig. 2.10.35: Overloaded Branch Key

After you change the overload percentages, you will have to go back to the kV color scheme and then back again to the overload color scheme in order to redisplay the map with the new percentages implemented.

## Command Dialog (View)

The Command Dialog box allows you to type Powerflow Command Language commands and send them to ipfsrv. It also allows you to see the communication that passes between the GUI and PF components of IPF as IPF runs.

---

**Note:** This command is intended for advanced users. It was created for program development and may be useful for users wanting to observe the interprocess communication channel traffic.

---

The top box is a scrolling text box that stores PCL commands you type in the text box labeled Command Entry. The PCL commands list can be double-clicked to put the command into the Command Entry box. The third box down shows you what IPF's GUI component sends across the IPC channel to the PF component. The fourth box down shows what PF sends back to the GUI in response to the previously passed command. The third and fourth boxes are output only and are not responsive to any mouse clicks. Note that you can vary the vertical size of both output-only boxes by pressing on the sash controls and moving them up or down.

---

**Note:** Be sure to terminate the command set with \*[EOM] or ^[EOM]. The former issues a synchronous command; the latter, an asynchronous command.

---

**Command Entry.** This text entry box is intended for valid PCL commands that you want to send to the PF component of IPF. There is no syntax checking at data entry time. PF does all the checking once a command is sent. The command is sent to PF when you press the Return key. See the *Powerflow Command Language (PCL)* section for information on valid commands.

**Close.** Clicking this button causes the dialog box to disappear from the screen. No other action is performed.

## Error Messages (Help)

When an error condition occurs, such as when IPF detects bad data, the Error Messages dialog box contains messages to help you determine what caused the error. See below.

These messages, along with many others, also appear in the terminal window where you started IPF. The other messages are usually not of any interest to a user. However, if you want to view the iteration report, you will have to look at the terminal window. This information is not displayed anywhere in the GUI.

---

**Note:** This dialog box pops up on its own only when the error message is a fatal one.

---

**Identifier.** This text box specifies the identifier (source code file) where an error condition occurred. This message assists the programming staff to locate the source message.

**Line.** This text box gives the source code line number associated with the identifier where an error condition occurred.

**Close.** This button closes the Error Messages dialog box.

**Help.** This button brings up the IPF help system window.

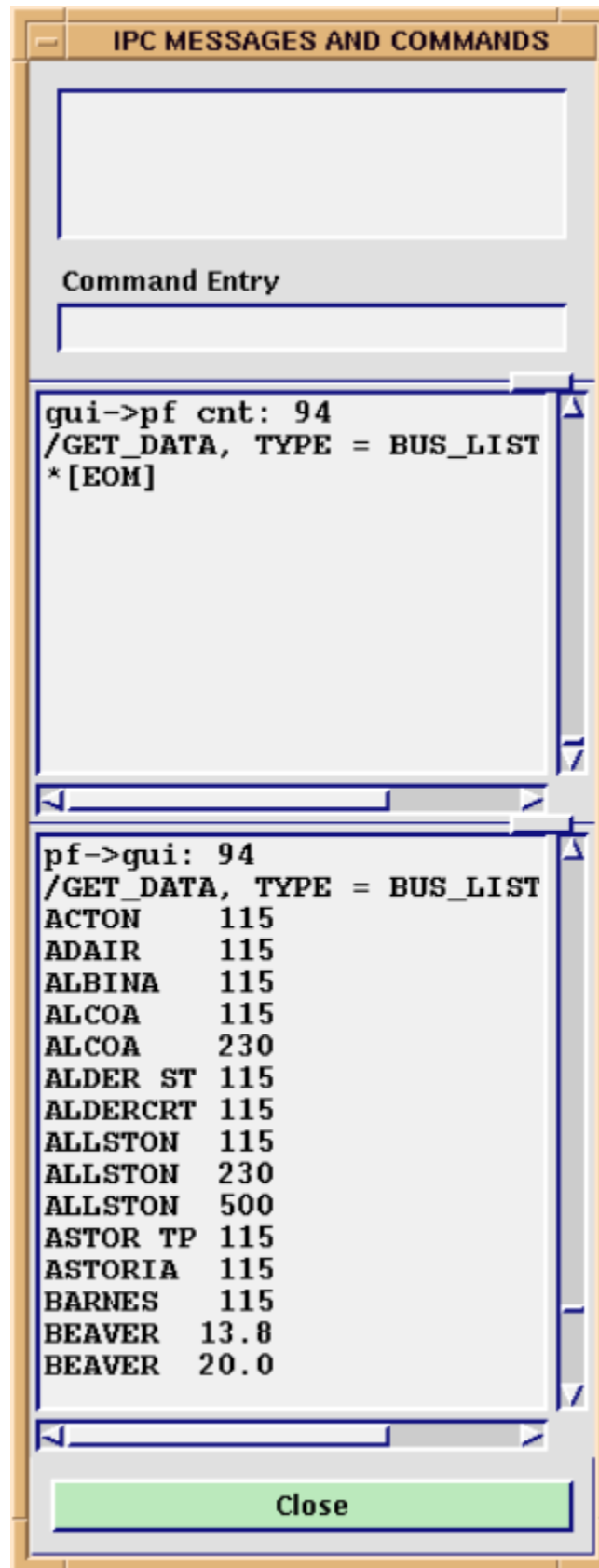


Fig. 2.10.36: Command Dialog Box

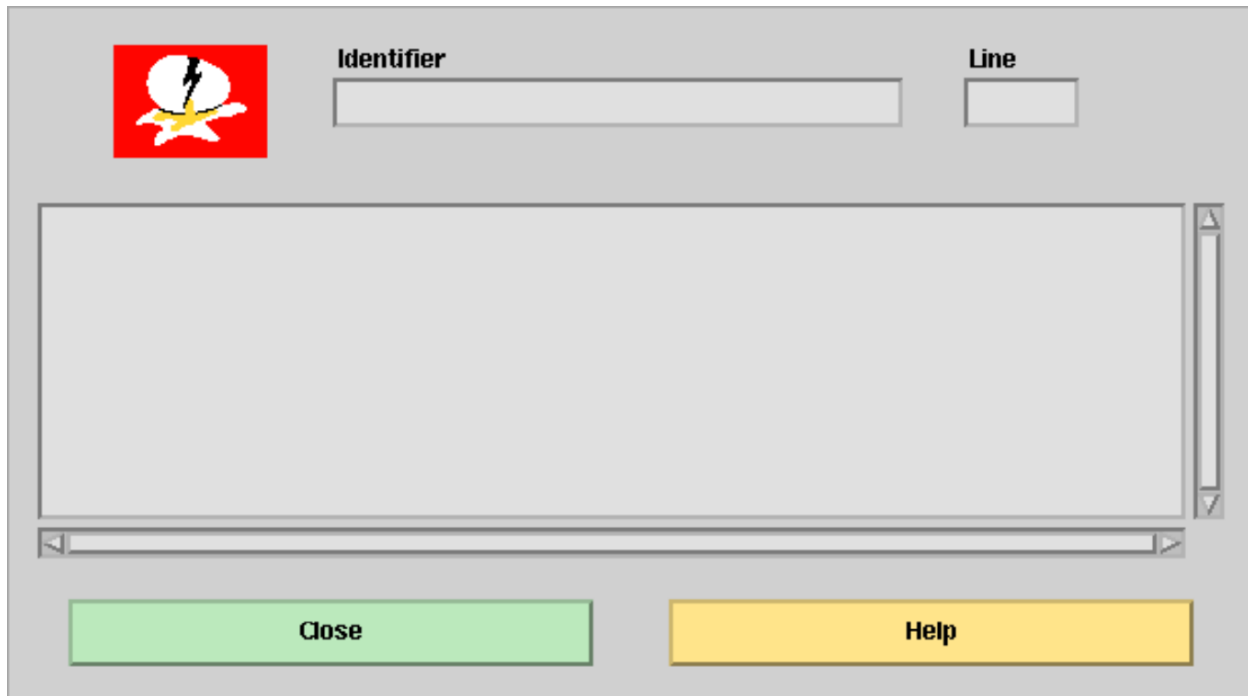


Fig. 2.10.37: Error Box Dialog Box

### Exit (File)

This command exits you from IPF. You can exit at any time. The exit command is also available on a button below the toolbox in the main window. Although you can also click on the X Windows menu button at the upper right of the window to close, this is not the recommended method of exiting from IPF. It should be used only as a last resort.

**OK.** Clicking this button exits you to the operating system.

**Cancel.** This closes the Exit dialog box without any action.

### General (Help)

The IPF help system provides a condensation of this documentation. See below.

In addition to the Page Up and Page Down window buttons and scroll bar, the Help system text itself has a feature to help you navigate. The text incorporates hyperlinks. These allow you to jump immediately to a desired spot in the help text.

At the top of the Help text is a list of topics. You can click on a topic to go to the text explaining that topic. To read the topic text on a page, use the scroll bar. Use the Page Up and Page Down buttons at any time to go to different pages, one page at a time.

The Annotate button allows you to attach your own notes to a particular page. See below. A red paper clip shows up in the left margin after you save a note, indicating that a note exists. To read an existing note, just click the Annotate button when you see a red paper clip.

**Page Up.** The Page Up button finds the next page marker toward the beginning of the IPF Help file. Page Up does not go to the top of the next page up. Use the scroll bar to scroll to the top of the page if necessary.

**Page Down.** The Page Down button finds the next page marker toward the end of the IPF Help file. Page Down does not go to the top of the next page down. Use the scroll bar to scroll to the top of the page if necessary.

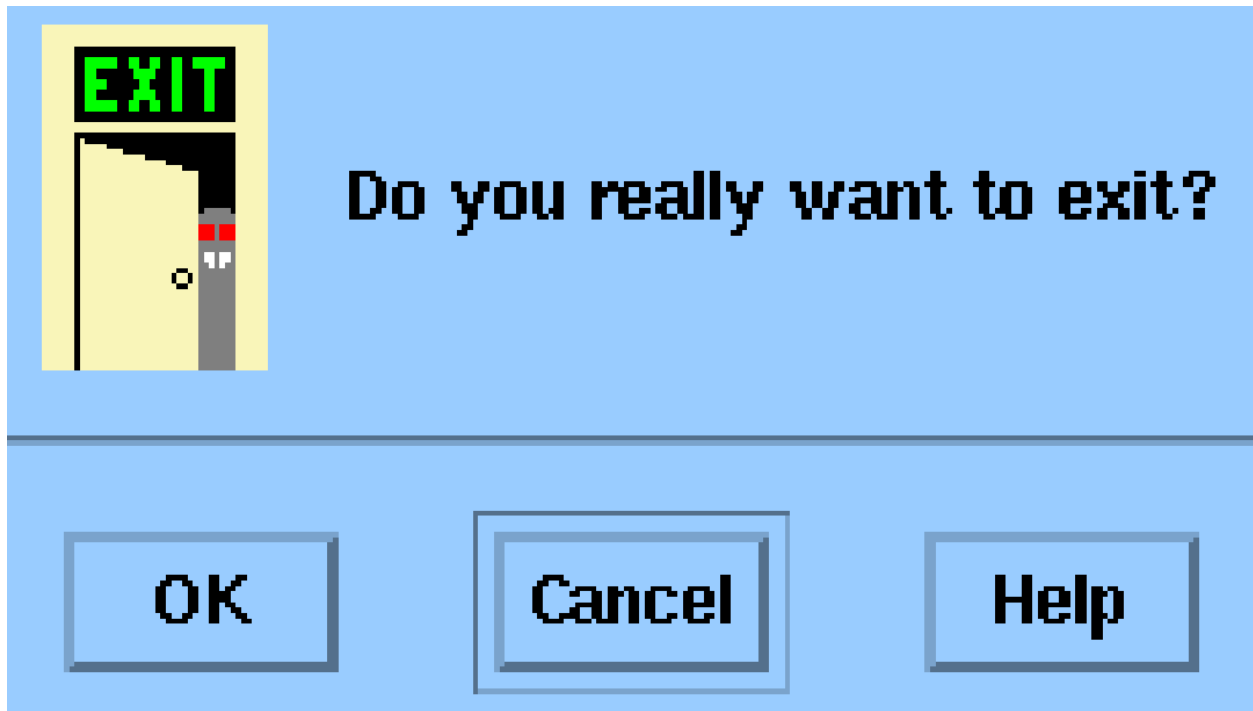


Fig. 2.10.38: Exit Dialog Box

**Annotate.** The Annotate button opens a text window that you can type text into. If you want to save the text, click the Save button; if you want to clear or remove the text, click the Remove button; Close closes the Annotation dialog box without making further changes.

**Close.** The Close button closes the Help system window.

### Network Data Edit (Edit)

There are two ways to edit bus and branch data in IPF. Editing via the Network Data Edit dialog box is just one. See below. The other one is the Input Data Edit Box. Each of the ways offers its own benefits. See the Input Data Edit Box for more information.

The advantage of Network Data Edit over the other two methods of editing is that you can access the entire network data base using filters. The other editing methods require accessing the network data through the displayed network diagram or through the Bus List dialog box.

---

**Note:** Network Data Editing is designed for the expert user. Editing a network data record directly is intended for advanced users who already know the exact columns for specific data fields in each network record. See *Powerflow Command Language (PCL)* for network record format information.

---

When you use the Network Data Edit dialog box, editing is performed indirectly on the resident network data using either an internal editor or an external editor. The internal editor is very basic; it permits cutting, pasting, and overtyping. The external editor can be any editor selected by the GUI setup script file. For UNIX systems, vi is usually the editor specified. There is no formatting or data validation support other than that provided by the user's external editor. No matter which you choose, the work is actually done on an intermediate file from which network changes may later be assimilated.





Fig. 2.10.39: General Help Dialog Box

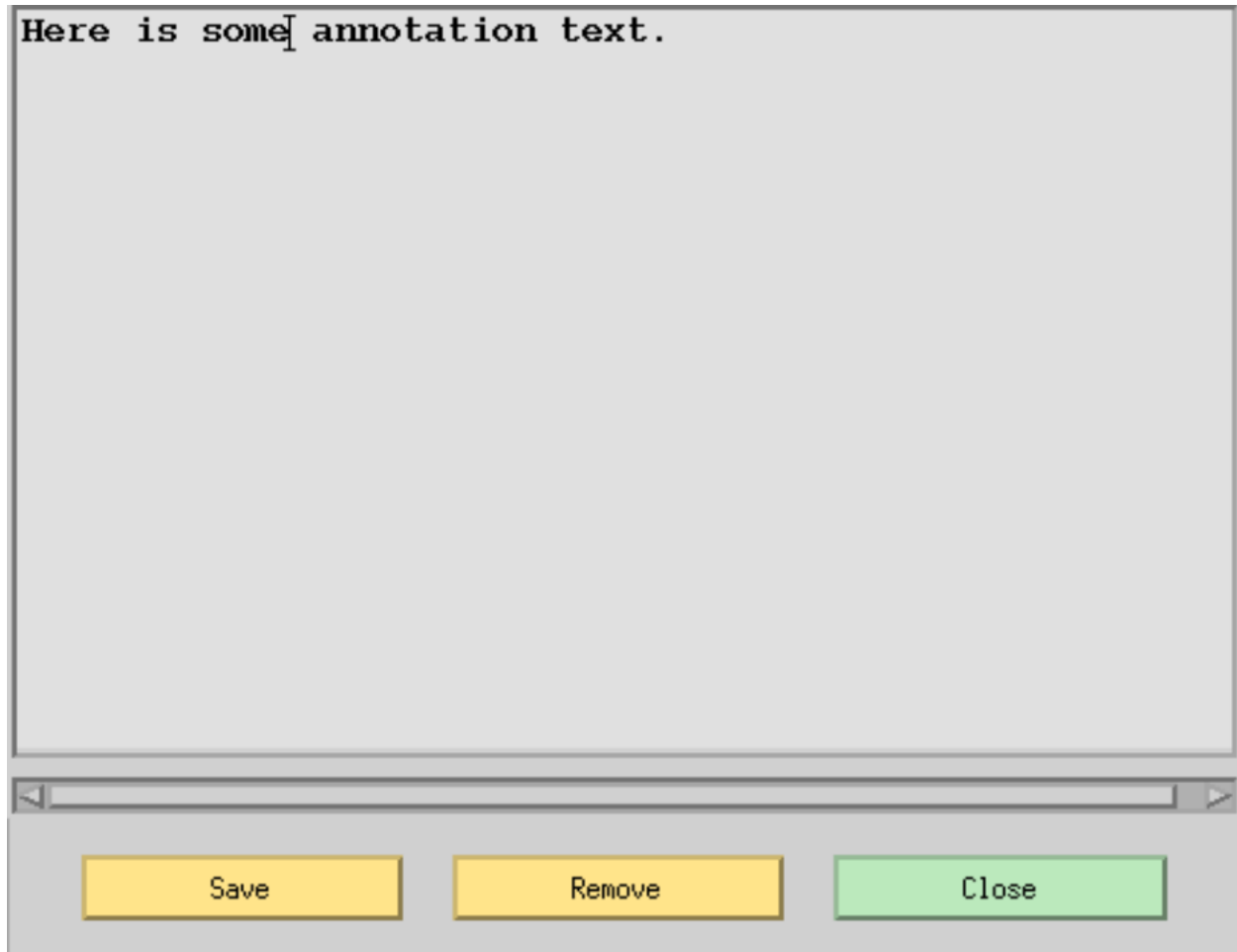


Fig. 2.10.40: Annotate Dialog Box

Editing of the network data is done with a screen editor. Four steps are required to successfully accomplish data editing. See below.

1. Select individual items within a filter. The filter list is scrollable so that you can see items outside the visible listing. When you click the left mouse button anywhere on a non-highlighted item, the line highlights by reverse video, indicating selection. You can unselect by clicking again on the same item using the left mouse button. You can continue to select as many additional items as you like. If none are selected, the default is that all the items for the filter are selected.
2. Apply the selected filter(s) by pressing the Apply button(s).
3. Click either the Internal or External Editor button to display the data.
4. Edit the data in the display box. You can use the cursor “arrow” keys to move around in the data and to cause it to scroll up and down.
5. Process the edited file using the Send To PF button.

The last step concludes the editing session. If it is skipped, no changes are performed upon the system. If it is applied, the edited file is automatically processed into network data changes and sent to Powerflow.

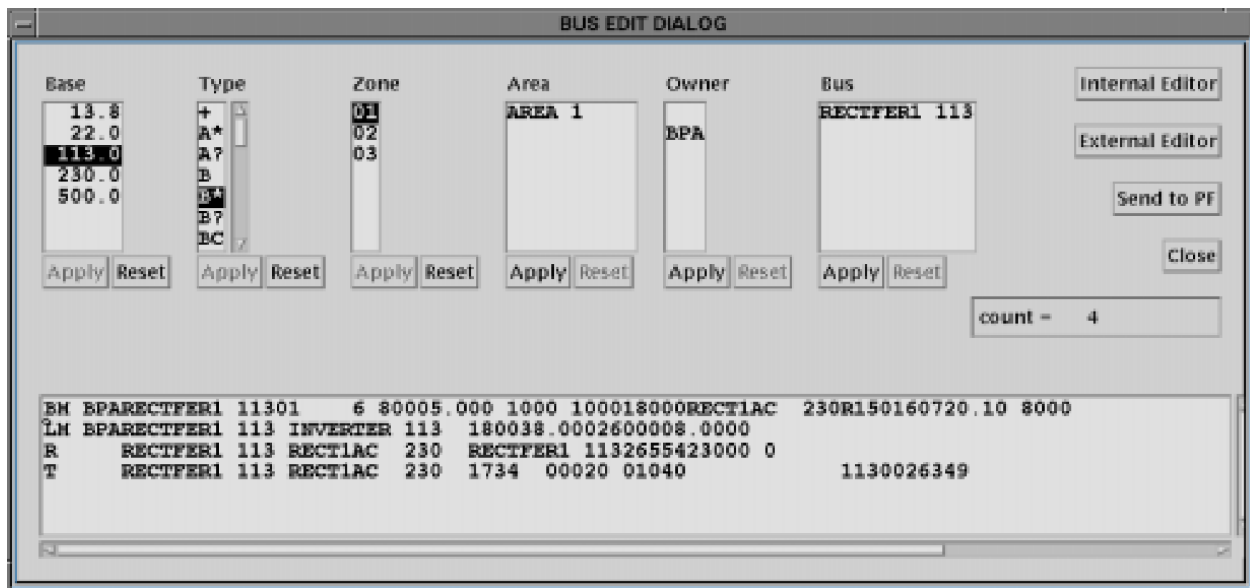


Fig. 2.10.41: Network Data Edit Dialog Box

## Dynamic Filters

The power of the Network Data Edit dialog box is realized in applying the dynamic filters. Six filters allow you to restrict the amount of network data displayed. The filters are Area, Base, Zone, Owner, Bus, and Type. The filters are dynamic because they propagate “downstream” (from left to right) the effects of previously defined filters upon the remaining filters.

Initially, all of the filters are primary and each filter displays the full attributes of its type. However, suppose you select one of the filters, say Owner, and highlight individual owners of interest. Then press the Apply button. Three things happen.

1. The Owner filter becomes the solitary primary filter; it is automatically repositioned to the leftmost position in the dialog box. The Apply button on this filter is ghosted, indicating that it has been applied.
2. The remaining filters are now collectively called secondary filters. They are repositioned on the dialog in an arbitrary order on the right of the primary filter. The effects of the primary filter are applied to the secondary filters through the network (downstreaming). There is one exception, Type, whose display is always static.
3. Only the network items that collectively meet the combined filter criteria are displayed.

The effects of the primary filter can be undone by pressing the Reset button. The procedure can be repeated for all remaining filters. The first selection, mentioned above, identifies the primary filter; the second, if selected, identifies the secondary filter; the third, the tertiary filter; etc. Select any of the remaining filters that do not have their Apply buttons ghosted. Make selections and press the Apply button. The filters will be repositioned again from left to right: primary, secondary, tertiary, etc.

The display for each filter is always current, and it shows selected network attributes (owners, zones, base kVs, etc.) that will be extracted if an Edit button is pressed.

Through judicious use of filters, you can reduce the size of the displayed network to a few hundred items. Since all of the displayed data must be encoded in Power flow and transported to the GUI process via the interprocess communication channel, the time to retrieve network data can become very long for large systems.

Table 2.10.2: Description of Type Filter

Code	Description
•	Retrieves all data.
A*	Retrieves A, A0, A1, A2, A3, A4, and I data.
A?	Retrieves A, A0, A1, A2, A3, A4 data.
I	Retrieves I data.
B*	Retrieves all bus and branch data.
L*	Retrieves all branch data including LTCs, transformers.
B?	Retrieves all bus types.
B, BE, BS, BC, BD, BV, BQ, BG, BT, BX, BM, BF	Retrieves specific bus type(s) named.
•	Retrieves all continuation bus data.
X	Retrieves X data.
Q	Retrieves QP, QX, and QN data.
L, LD, LM, E, T, TP, R, RZ	Retrieves specific branch type(s) named.

**Apply.** Clicking this button applies the selection of items within the filter to the remaining filters. It also promotes that filter to the next available downstream order: primary, secondary, tertiary, etc.

**Reset.** Clicking this button resets the filter in context with its status (primary, secondary, tertiary, etc.). If you wish to reset the filter to its primary state, you must first reset the original primary filter - the leftmost displayed one. Resetting a filter also unhighlights any prior selected items.

**Internal Editor.** Clicking this button applies the filters to the network database, and displays the retrieved data in the editing portion of the dialog box. The edit dialog box is a scrollable list; it supports only basic functions, such as cutting, pasting, and overtyping. The input data for the editor resides in the file `editbus.dat`; the output data is in the file `editbusn.dat`.

**External Editor.** Clicking this button applies the filters to the network database, and displays the retrieved data in a selected external editor in a new terminal window. For UNIX, the default external editor is `vi`. The input data for the editor resides in the file `editbus.dat`; the output data is in the file `editbusn.dat`.

**Send To PF.** Clicking this button applies a script file to the two files `editbus.dat` and `editbusn.dat`, translating the differences into a change file `editbusc.dat`. Network data changes are not made until this step is performed.

**Close.** Clicking this button closes the dialog box. This does not cancel any changes which have already been sent to PF.

## Reviewing Network Changes

Once the Send To PF button is clicked, the editing changes are translated into equivalent Power flow data change records, which are sent to Powerflow. Since little or no data checking or validation is performed prior to the dispatch, the possibility of errors is high. You can check the status of your changes by viewing the Network Changes report in the View-Reports menu and the System Errors in the Help menu.

## Notes

When using the internal editor, remember that you are in *overtyp*e mode only. Delete and Backspace keys may delete forward or backward depending on your X defaults. The Tab key inserts a blank character, but there is no insert mode.

The Powerflow and GUI must be launched from the same directory and reside on the same platform. The file passed in the IPF command `/CHANGES, FILE = editbusc.dat` does not contain directory or node prefixes.

The branch data is displayed double entry. If both terminal buses of a mutual branch are displayed in the edit dialog box, then the branch is displayed twice. It suffices to edit either branch entity; it is redundant to edit both branch items.

Editing an identification field amounts to deleting an old network data entity and adding a new network data item. Editing a data field amounts to modifying the data field. There is a special problem with blanks in data fields. Blanks, when applied to change modifications, designate “no changes.” You must enter a zero (0) to delete a data field within a network data record.

An external program `ipf_diff` translates differences between files `editbus.dat` and `editbusn.dat` into equivalent network data changes and puts them into the file `editbusc.dat`. The script defining this procedure must be installed in the GUI.

## Open (File)

The Open command allows you to specify which files you want to work with. You can open files, work with them, and save them (or discard the work and not save them) as many times as you like, once IPF is executing. You do not need to exit IPF and start it up again once you are done with a given set of files. You simply load the next file(s) you want to work with. They replace (overwrite in memory only!) the previously loaded files. See *Save and Exit*.

When you first choose Open, you may see file names already existing in some of the text boxes. These default file names come from the X resource file for IPF, the XGUI file, in your logon directory. You can change these defaults to fit your operation. Read about the XGUI file in [Customizing the GUI \(XGUI\)](#).

There are three areas in the OPEN dialog box: a file type selection panel at the left, a standard Motif file selection dialog in a panel at the right, and a “load files” area at the bottom. See below.

The basic file specifying and loading process goes like this:

1. Select a file type in the file type area.
2. Specify the exact file to load in the file selection dialog.
3. Do steps 1 and 2 until all files you want to work with are selected.
4. Load them all with the Load Selection button at the bottom.

All text boxes where file names appear are standard Motif file name text boxes. Thus, you can modify the file names at any time simply by typing new text into the text box. If a full name is hidden at one end or the other of a text box, put the mouse cursor in the text box, press the left mouse button and drag till the hidden file name text comes into view. Press and ‘paint’ to select text; then type to replace it.

The asterisk (\\*) in a file name is a “wildcard” character that stands for “any string of characters of arbitrary length.” So, for example, `*.BSE` matches any file name that ends in `.BSE`.

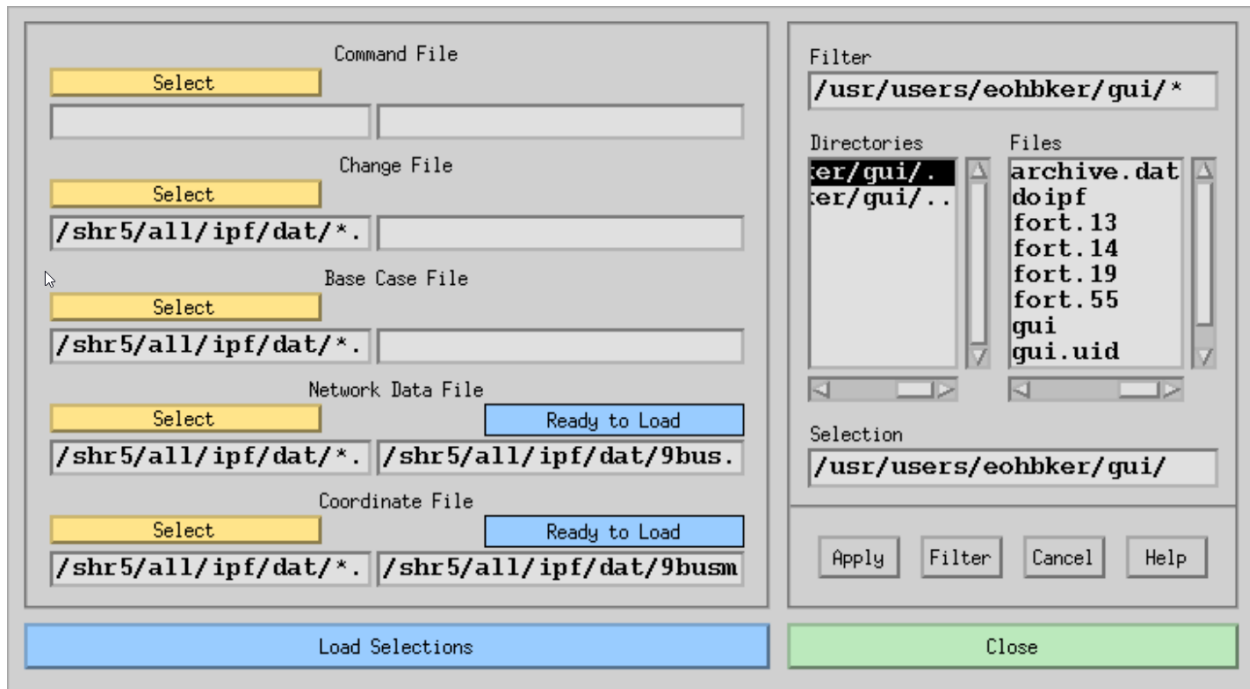


Fig. 2.10.42: Open Dialog Box

**Command File.** One of five areas in the file type selection panel. The associated Select button and file text boxes apply to IPF command files. This ASCII text file is made up of standard Powerflow Command Language (PCL) commands and data. PCL files are generally used by advanced users. See the *Powerflow Command Language (PCL)* for more information.

**Change File.** One of five areas in the file type selection panel. The associated Select button and file text boxes apply to IPF change files. This ASCII text file is made up of commands and data records representing modifications intended for the current case under study. Such a file is applied automatically when it is loaded.

**Base Case File.** One of five areas in the file type selection panel. The associated Select button and file text boxes apply to IPF base case files. This is the main, binary format ‘history’ file of IPF after it has run a solution.

**Network Data File.** One of five areas in the file type selection panel. The associated Select button and file text boxes apply to IPF “raw” network data files. This ASCII text file consists of bus and branch data and usually represents basic data for a new network to be studied with IPF.

**Coordinate File.** One of five areas in the file type selection panel. The associated Select button and file text boxes apply to IPF coordinate files. This ASCII text file contains position and identification data for buses and branches in a given network. The file may be modified independently of its associated base case file or network data file.

**Select.** A button whose action transfers its associated file name filter with the wildcard (\*) to the file name text box labeled Filter in the file selection dialog. Pressing Select causes the file selection dialog box to change to the directory specified by the wildcard file name in the file name text box associated with the Select button. The filter is specified in the left-hand text file box. The right-hand text file box contains the name of the actual file to be loaded, *not a file name filter*.

**Ready to Load.** Feedback information telling you the current state of the file in the associated file name text box. *This is not a button*. “Ready to Load” reminds you that IPF does not yet know about this file. You must load it with the Load Selections button below.

**Loaded.** Feedback information telling you that you have previously loaded the associated file. Thus, IPF is ready to operate on this file. **This is not a button**. If there was some problem loading the file, this box will say “Not Loaded”.

**Filter.** The file name text box contains a “filter” that selects categories of file names. You can directly modify the file name text by selecting text and typing in new text, or more commonly, you can use the Select button in the file type selection panel at the left to put a file name filter in the box. If you modify the file name filter, you must click the Filter button below to change the directory and file list displays.

**Directories.** This list component contains directory names associated with the filter directory. You cannot modify these directory names by selecting them and typing. Use the scroll bars at the side and bottom to move the directory names up and down or back and forth. Click the arrows or the click the bar between the arrows and drag to move the names so you can see them. You move up the directory structure by double clicking the directory name ending with a period-period (.). You move down by selecting the name of the directory you want to move into. You can also select a directory and press the Filter button.

**Files.** This list component contains file names that are within the directory named at the left that ends with a period (.) and that satisfy the filter criterion specified above in Filter. You cannot modify these file names by selecting them and typing. Use the scroll bars at the side and bottom to move the file names up and down or back and forth. Click the arrows or click the bar between the arrows and drag to move the names so you can see them.

You select a file name by clicking once on the file name. This puts the selected file name in the Selection file name text box below. You can then put the selected file name in the appropriate file name text box in the file type selection panel at the left by pressing Apply. You can accomplish the same thing by double clicking the file name in the Files box.

**Selection.** This file name text box receives the file name selected by clicking a file name in the Files file list. Or, since it is a text box, you can directly modify the file name by selecting text and typing replacement text. Be sure to type an exact file name; do not use a wildcard character such as the asterisk (\*) as part of the file name.

**Apply.** This button causes whatever file name is in the Selection text box to be put in the appropriate file name text box in the file type selection panel at the left. This has the same effect as double clicking the file name in the Files file list.

**Filter.** Clicking this button causes all file names satisfying the filter file name text in Filter to appear in the Files file list. It also puts the currently selected directory name *without the file name* in the Selection text box. You must select a file name from the Files file list to select a specific file name.

**Help.** (Not yet implemented).

**Load Selections.** Clicking this button sends a “load all the specified files” message to the PF component of IPF. The dialog box also closes and disappears. You can verify that the specified files were loaded by selecting OPEN again. Files that have been previously loaded display the “Loaded” message above the file name text box.

**Close.** Clicking this button closes the dialog box without changing any directory or file name values. You can close at any time.

---

**Note:** For more information about the various file types you can open, see [Network Data](#).

---

## PF ID/Description (Edit)

The PF ID/Description dialog box is the way you provide titling and comments for a case. These are saved with the case when you save it in a base case file with the Save command. If you loaded a case from a base case file, any titling and comments that were in the file will be displayed in this dialog. See below.

**Case Id.** Ten character maximum, alphanumeric field identifying the base case.

**Description.** Twenty character maximum, alphanumeric field identifying the particular study, project, etc.

**PF Header(s).** These three records will be printed on each batch output report page and hardcopy map. The first line is generated by the program and contains the Case ID and Description fields. The other two lines can be entered by the user.

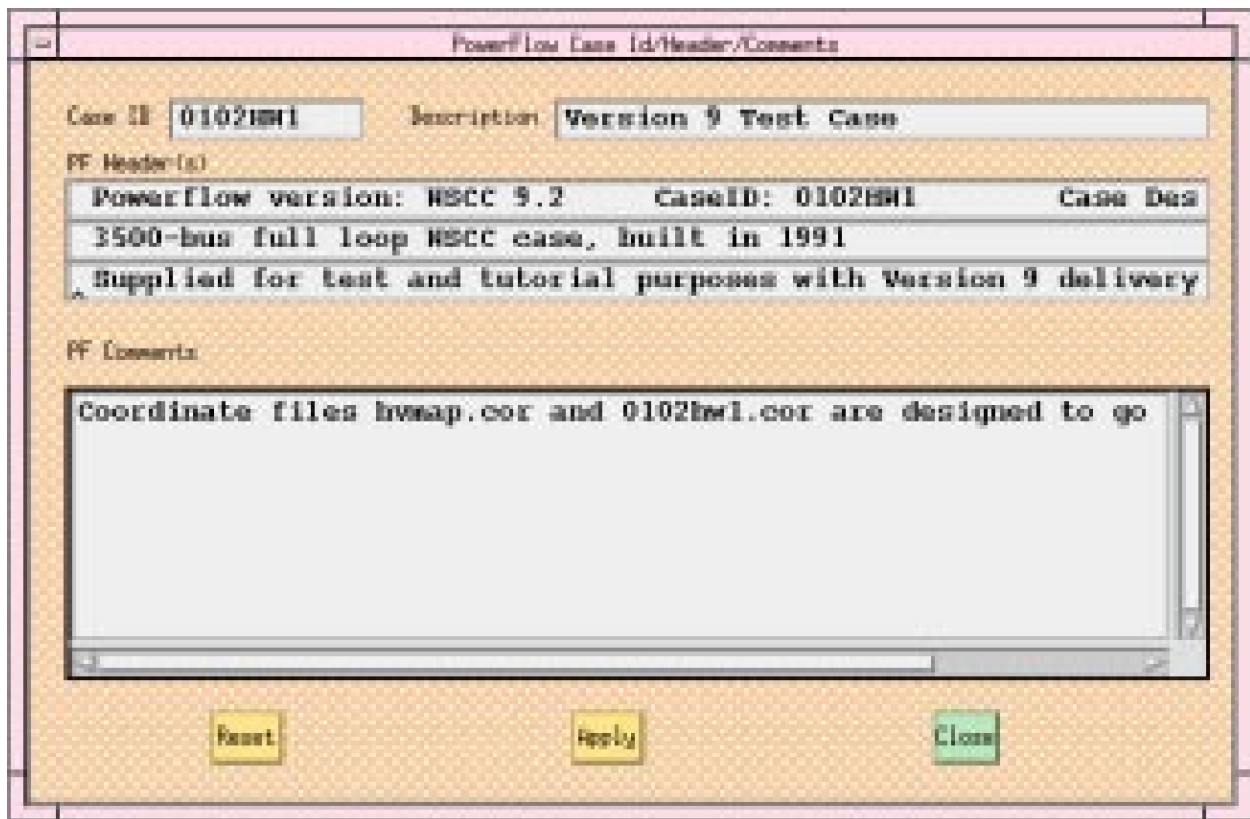


Fig. 2.10.43: PF/ID Description Dialog Box



**PF Comments.** Up to 20 additional lines of description and comment can be added. These are stored with the system data, so they will be visible here when you open an old base file. They are also printed in conjunction with certain batch reports, if you load the old base in BPF.

**Reset.** Clicking this button erases the Case, Description, and PF Comments text boxes, and closes the dialog box.

**Apply.** Clicking this button sends the current contents of all the text boxes to the currently resident case and closes the dialog box. If you save the case, everything you have applied will be saved with the case. The dialog will show any saved text when it is opened.

**Close.** Clicking this button closes the dialog box without changing any values in any of the text boxes. You can close at any time.

## Plot Options (File)

The Plot Options command allows you to create a network diagram, or map file that can be printed on a PostScript compatible printer. The plots may have insets and legends as well as the usual power system bus, branch, generator, shunt, series compensation, area bubble, intertie line, and other symbols. They are not a copy of what you see on the screen, although both the display and the hard copy are based on the coordinate file you currently have loaded.

Four dialog boxes, User Comments, Page Options, Diagram Options, and Plot Destination, allow considerable customization of the default power system plot. The basic diagram layout is selectable from the Diagram Options dialog box. In addition to the default of real and reactive power flow, current and MVA, loss, difference, and interchange diagrams are selectable.

Some computer systems provide previewing capability on your display by means of a separate PostScript viewing utility. Check with your system administrator to see if your system has this capability.

See [Network Diagrams](#) for information about how IPF produces a PostScript plot file.

## User Comments

The User Comments menu item provides a way for you to include text comments on a plot. See below.

You can enter comments that will be shown in a block on the diagram. A comment beginning with an ampersand (&) identifies an auxiliary coordinate file name. Only one such file is currently allowed, and it must be identified in the last comment. The auxiliary coordinate file data will be appended to the coordinate file data edited in the GUI

**OK.** Clicking this button accepts all text as it is currently shown, passes this on to IPF, and closes the dialog box.

**Plot Now.** Clicking this button sends the currently resident plotting information to the currently selected printing device, and then closes the dialog box.

**Close.** Clicking this button closes the dialog box without changing any text you have entered. No text is passed on to IPF. You can close at any time.

## Page Options

The Page Options dialog box facilitates selection of options that control the general appearance of a network diagram. The options you choose are incorporated into the network diagram file. See below.

**Orientation.** Portrait specifies that the long axis of the paper is vertical. Landscape specifies that the long axis is horizontal. Portrait is the default.

**Transparency.** Insets such as the legend or a detailed section of the diagram can show what is underneath them. This property is called *transparency*. Opaque specifies that objects underneath will *not* show through. Transparent specifies

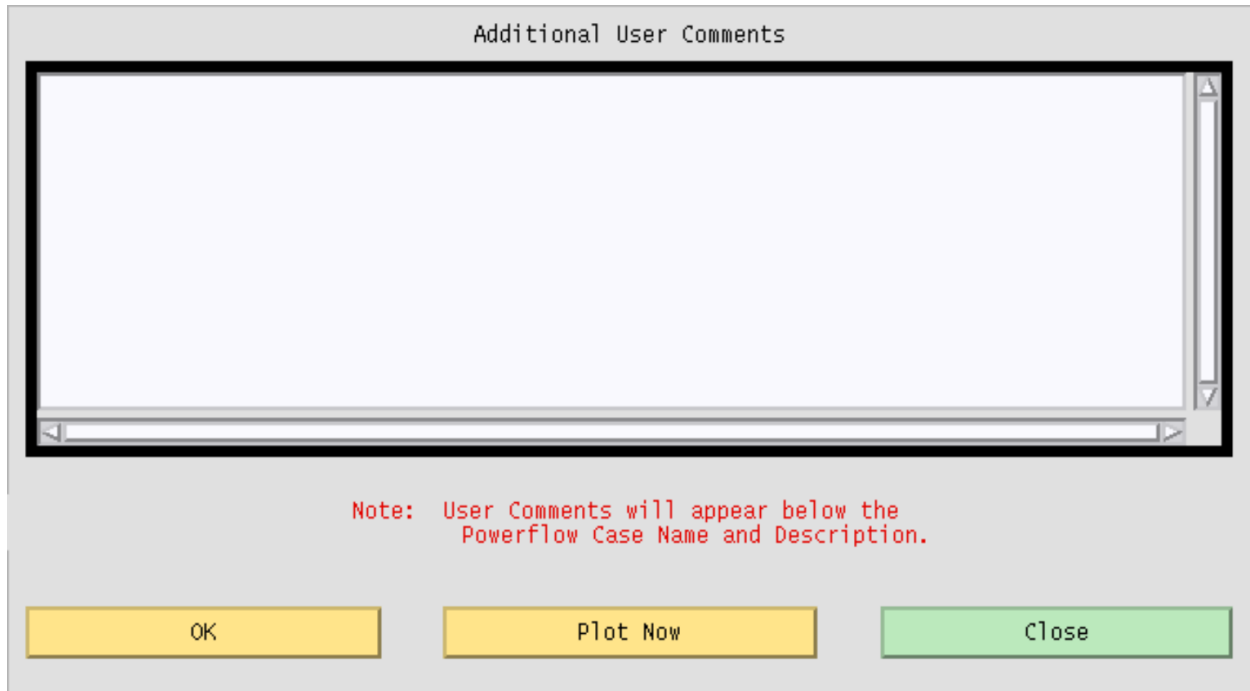


Fig. 2.10.44: User Comments Dialog Box

that objects underneath *will* show through. For insets, legends, and identification labels usually Opaque is specified. Opaque is the IPF default.

**Paper Size.** Width specifies the width of the paper. Height specifies the height of the paper. Make sure your printer can handle the size paper you specify. Two buttons specify 21.59 cm by 27.94 cm (8.5 by 11 inch) paper or 27.94 cm by 43.18 cm (11 by 17 inch). The IPF default is 21.59 cm by 27.94 cm (8.5 by 11 inch).

**Border Top Right Corner.** A border is a solid-line rectangle surrounding graphical objects. X cm and Y cm are real numbers that specify x and y values for position with respect to the top right corner of the drawing area. The lower left corner is analogous to (0,0) in the Cartesian coordinate system.

**Case Name Position.** A case name is the user-specified identification associated with a particular case study and is saved in the base case file. X cm and Y cm are used to position the case name.

**Comments Position.** Comments are from the User Comments dialog box and the base case file. Offset. X cm and Y cm are real numbers specifying the x and y locations of the lower left corner of an insert relative to the lower left corner of the paper. The lower left corner is analogous to (0,0) in the Cartesian coordinate system. The default values for X cm and Y cm are 0.0.

**Scale Factor.** X and Y are real numbers specifying whether the x and/or y axes should be enlarged or reduced. Numbers larger than 1.0 enlarge the size of diagram objects. Positive numbers smaller than 1.0 reduce the size of diagram objects. The default values for X and Y are 1.0.

**Label Box Coordinates.** Typically, the label box is in the lower right corner of the diagram. You may then enter just the top left corner of the box. The bottom right corner must be entered also if the box is elsewhere in the diagram.

When the label box option is selected, several default locations are established relative to the upper left corner of the box. Default options, which can be overridden by options on the Page Options menu, are

- Coordinate file name, positioned above the box.
- Powerflow case name, date, and Powerflow program version, positioned inside the box.
- Powerflow description, positioned below the Powerflow case name.

Orientation		Transparency		Paper Size	
<input checked="" type="radio"/> Portrait		<input type="radio"/> Opaque		Width	<input type="text" value="21.59"/>
<input type="radio"/> Landscape		<input checked="" type="radio"/> Transparent		Height	<input type="text" value="27.94"/>
Border Top Right Corner		Case Name Position		<input type="text" value="21.6 x 28 cm"/>	
X cm	<input type="text" value="21.59"/>	X cm	<input type="text" value="12.7"/>	<input type="text" value="28 x 43 cm"/>	
Y cm	<input type="text" value="27.94"/>	Y cm	<input type="text" value="4.0"/>		
Comments Position		Offset		Scale Factor	
X cm	<input type="text" value="12.7"/>	X cm	<input type="text" value="0.0"/>	X	<input type="text" value="1.0"/>
Y cm	<input type="text" value="3.7"/>	Y cm	<input type="text" value="0.0"/>	Y	<input type="text" value="1.0"/>
Label Box Coordinates (Centimeters)					
<input type="text" value="12.0"/>		<input type="text" value="4.5"/>	<input type="text"/>	<input type="text"/>	
TOP LEFT CORNER		BOTTOM RIGHT CORNER			
Legend Position					
X cm	<input type="text" value="5.0"/>	Y cm	<input type="text" value="0.5"/>		
<input type="button" value="OK"/>			<input type="button" value="Close"/>		

Fig. 2.10.45: Page Options Dialog Box

- Comments entered by the user, positioned below the Powerflow description.
- Comments from the Powerflow case, positioned below comments entered by the user.
- Border; the maximum size allowable considering the paper size and printer capabilities.

Accounting information is below the border and is fixed. This information shows the diagram type, flow options, and dates and versions of the Powerflow program used to generate the Powerflow case and diagram.

**Legend Position.** This locates the top left corner of the legend. The legend provides a description of branch symbols used in the diagram.

**OK.** Clicking this button accepts all values as they are currently set and closes the dialog box.

**Close.** Clicking this button closes the dialog box without changing any values. No changed values are passed on to IPF. You can close at any time.

### Diagram Options

The Diagram Options dialog box allows for selection of power flow values that appear on a network diagram. The options you choose determine which power flow solution data is combined with coordinate data to create a PostScript network diagram file. See below.

**Bus Name.** Coordinate files contain the bus record identifier consisting of bus name and base kV and an alternate coordinate diagram identifier called a bus abbreviation. The Abbreviation identifier simplifies crowded diagrams. Abbreviation is the default.

**Bus Voltage.** Bus Voltage allows either the actual kV or per unit voltage to be displayed. The actual kV value is the default.

**Generation.** Specifying Generation means that the P and Q generation values and generator graphic symbol show on the diagram.

**Shunt.** Specifying Shunt allows you to see shunt values and a graphic symbol.

**Show Phase Angle.** Specifying Show Phase Angle means that bus angle with respect to the slack bus shows on the diagram.

**Show Load.** Specifying Show Load allows you to see actual load values near the bus voltage values.

**Show Total Flow of Undrawn Branches.** Specifying Show Total Flow of Undrawn Branches allows you to see P and Q flow values into undrawn branches.

**Show Outages.** Specifying Show Outages means that outaged branches show in the diagram.

**Parallel Lines.** Specifying Combined adds multiple circuit power flow values together. Specifying Separate allows you to see flow values for each individual branch. Combined is the default.

**Transfrmr Taps.** Transfrmr Taps shows any transformer taps.

**Compensation.** Compensation shows series compensation values.

**Outages.** Outages shows outaged branches.

**Values.** Specifying Normal shows the solution values of a given Powerflow case. Specifying Difference allows you to compare two power flow cases and show the differences.

**Diagram Type.** Diagram Type can be only one type at a time. Specifying PQ Flow shows P and Q power flow values. This is the default. Specifying MVA & I shows megavoltamperes on transformers and current flows on lines. Specifying Loss shows transmission system loss values. Specifying Interchange creates a diagram that shows net generation, load, and line flow values for area interchange studies. Specifying Coordinates creates a diagram with no solution data. This may be used for verifying that bus and branch placement is satisfactory.

Bus Detail	Branch Detail	Diagram Type
<p><b>Bus Name</b></p> <p><input checked="" type="radio"/> Abbreviation</p> <p><input type="radio"/> Full Name &amp; KV</p>	<p><b>Parallels</b></p> <p><input checked="" type="radio"/> Combined</p> <p><input type="radio"/> Separate</p>	<p><input checked="" type="radio"/> PQ Flow</p> <p><input type="radio"/> MVA &amp; I</p> <p><input type="radio"/> Loss</p> <p><input type="radio"/> Interchange</p> <p><input type="radio"/> Coordinates</p> <p><input type="radio"/> Loading %</p>
<p><b>Bus Voltage</b></p> <p><input checked="" type="radio"/> kV</p> <p><input type="radio"/> Per Unit</p>	<p><input type="checkbox"/> Transfmr Taps</p> <p><input type="checkbox"/> Compensation</p> <p><input checked="" type="checkbox"/> Outages</p>	
<p><b>Generation</b></p> <p><input checked="" type="radio"/> Draw Generation as Needed</p> <p><input type="radio"/> Always Draw Generation</p> <p><input type="radio"/> Draw No Generation</p>	<p><b>Values</b></p> <p><input checked="" type="radio"/> Normal</p> <p><input type="radio"/> Difference</p>	<p><b>Flow Detail</b></p> <p><input checked="" type="checkbox"/> P Sending End</p> <p><input checked="" type="checkbox"/> Q Sending End</p> <p><input type="checkbox"/> P Receiving End</p> <p><input type="checkbox"/> Q Receiving End</p>
<p><b>Shunt</b></p> <p><input checked="" type="radio"/> Draw Shunt as Needed</p> <p><input type="radio"/> Always Draw Shunt</p> <p><input type="radio"/> Draw No Shunt</p> <p><input checked="" type="checkbox"/> Show Phase Angle</p> <p><input type="checkbox"/> Show Load</p> <p><input type="checkbox"/> Show Total Flow of Undrawn Branches</p> <p><input checked="" type="checkbox"/> Show Outages</p>		
	<p>OK</p>	<p>Close</p>

Fig. 2.10.46: Diagram Options Dialog Box

**Flow Detail.** The options in Flow Detail allow you to fine tune how you want to show the P and Q flow values. Your choices are: P Sending End, Q Sending End, P Receiving End, and Q Receiving End. P Sending End and Q Sending End is the IPF default.

**OK.** Clicking this button accepts all values as they are currently set and closes the dialog box.

**Close.** Clicking this button closes the dialog box without changing any values. No changed values are passed on to IPF. You can close at any time.

### Plot Destination

The Plot Destination option allows you to select the printer device most convenient or suitable to your job. You select a print command from the top list window, and the command appears in the text box below. You can edit the command in the text box. See below.

When you first open this dialog box, a default print command shows in the Selection text box. IPF reads the XGUI file in your home directory to find this print command. You can edit the ASCII text XGUI file to change this default print command. See *Customizing the GUI (XGUI)* for a discussion of how to modify the XGUI file.

**Items.** This scrolling window shows the print plot commands currently available. Selecting an item in the window makes it appear in the Selection text box below.

**Selection.** This text box contains the command string that IPF uses to print a diagram. You can modify the text in this box any way you like, perhaps to specify a printer or printer command option you use only occasionally. IPF uses whatever printer command is in this text box to print a diagram.

**OK.** Clicking this button stores the command string in the Selection text box to be used for subsequent printing. The Plot Destination dialog box then closes. Use the Print Plot command to actually send a diagram to the printer. Diagrams either printed or not printed are available for on-line viewing if you have a PostScript previewer on your system.

**Cancel.** Clicking this button closes the dialog box without changing any values. No values are passed on to IPF. You can cancel at any time.

**Help.** (Not yet implemented).

### Print Plot (File)

The Print Plot command sends the current, memory-resident base case/coordinate file data to the printer you have currently designated, using the settings specified in the Plot Options command. See the Plot Options entry in this section.

### Reports (View)

Reports are data extracted from a memory-resident base case that is then formatted for output to the display. In a few cases, data is extracted from a base case file that you specify. IPF supplies the most frequently needed reports via the Reports dialog boxes. See figures below.

The scrolling list of report possibilities appears in the upper portion of the Reports dialog boxes. You choose one of the report types by clicking on it. The dialog box itself then changes to reflect the kind of report you have chosen. Refer to the figures in this section to see a few of the possibilities. Generally, once you have chosen the kind of report you want, you then use the filter lists of Areas, Zones, Owners, etc., to narrow down the information you are looking for. Then you click View Report to see all the bus or branch records your specified list of criteria produces, or Save or Append Report to send the output to a file.

There is a limit of ten items selected on any filter. You can choose more than this, but only the first ten will be applied and reported. If you want *all* of any filter list, just *do not select anything* in the list (e.g. all areas, all zones in an area, etc.).



Fig. 2.10.47: Plot Destination Dialog Box

---

**Note:** The CFLOW library is available to C programmers so that more extensive reports can be extracted from IPF data. See [CFLOW C API \(libcflow\)](#) for more details.

---

**Reports Available.** This scrolling list shows all reports available. You select a report by clicking on it. If the report needs another base case file to do a comparison, a files menu will pop up so you can select the other file names.

**Area.** Click on the *areas* you want to report. Click again to unselect. Selections across Areas, Zones, Owners, Base kV, and Types are effectively “anded” and therefore narrow the reported information.

**Zone.** Click on the *zones* you want to report. Click again to unselect. Selections across Areas, Zones, Owners, Base kV, and Types are effectively “anded” and therefore narrow the reported information.

**Owner.** Click on the *owners* you want to report. Click again to unselect. Selections across Areas, Zones, Owners, Base kV, and Types are effectively “anded” and therefore narrow the reported information.

**Base kV.** Click on the *base kVs* you want to select. Click again to unselect. Selections across Areas, Zones, Owners, Base kV, and Types are effectively “anded” and therefore narrow the reported information.

**Type.** Click on the *bus types* you want to select. Click again to unselect. Selections across Areas, Zones, Owners, Base kV, and Types are effectively “anded” and therefore narrow the reported information.

**File Report Name.** The file name where you want report data saved. If no name is specified, the report will be saved in REPORTS.DAT.

**View Report.** Clicking this button brings up the View Report dialog box. The report data is in a scrolling text window. See the examples below. The number of lines of output is limited. If you see “MORE” at the bottom of the report, you will know it was truncated.

**Save Report.** Clicking this button saves the specified report to a file that you name in the File Report Name box. The file will contain all lines of the report, even if there are more than can be displayed on the screen.

**Append Report.** Clicking this button adds the report data to an already existing file. You must supply the file name you want to append to.

**Close.** Clicking this button closes the dialog box without saving any settings or information.

**Search Bus Name KV.** You type in this text box to go to a specific bus. The search function is case sensitive; that is, a and A are not the same. The search function begins as soon as you type a character. Begin typing the first characters of a bus name and the search function finds the alphabetically first bus matching the letters typed so far. For example, A finds ACTON, AL finds ALBINA, ALD finds ALDER ST, and ALDERC finds ALDERCART.

Clicking on a bus, or finding it with the search, causes it to be highlighted (selected). *All highlighted buses will be reported*, (up to the limit of ten). To unselect a bus that you do not want reported, just click it again.

**Limits.** Under % Line Load you can specify a minimum percentage for reporting overloads on line and transformers by typing a value in this text box.

Under PU Volt Relax you can relax the limit for over/under voltage reporting. The normal limits are either the global ones or those specified on the area record. If these give you too many buses, you can use this text box to extend the limit. For example, entering 0.02 will cause to be reported only buses whose voltage is more than 0.02 beyond the limits (you cannot lower the limit by entering a negative number.)



Reports available - choose one

**Bus Input Data**

Named Bus Bus/Branch Input Data

Named Bus Bus/Branch Output Data

Overloaded Line

Overloaded Transformer

Overvoltage/Undervoltage

Area	Zone	Owner	Base kV	Type
AREA 1	01		13.8	B
AREA 2	02	BPA	22.0	BC
AREA 3	03	LAP	113.0	BD
			230.0	BE
			500.0	BF
				BG

File Report Name

**View Report** **Save Report** **Append Report** **Close**

Fig. 2.10.48: Reports Dialog Box: Bus Input Data

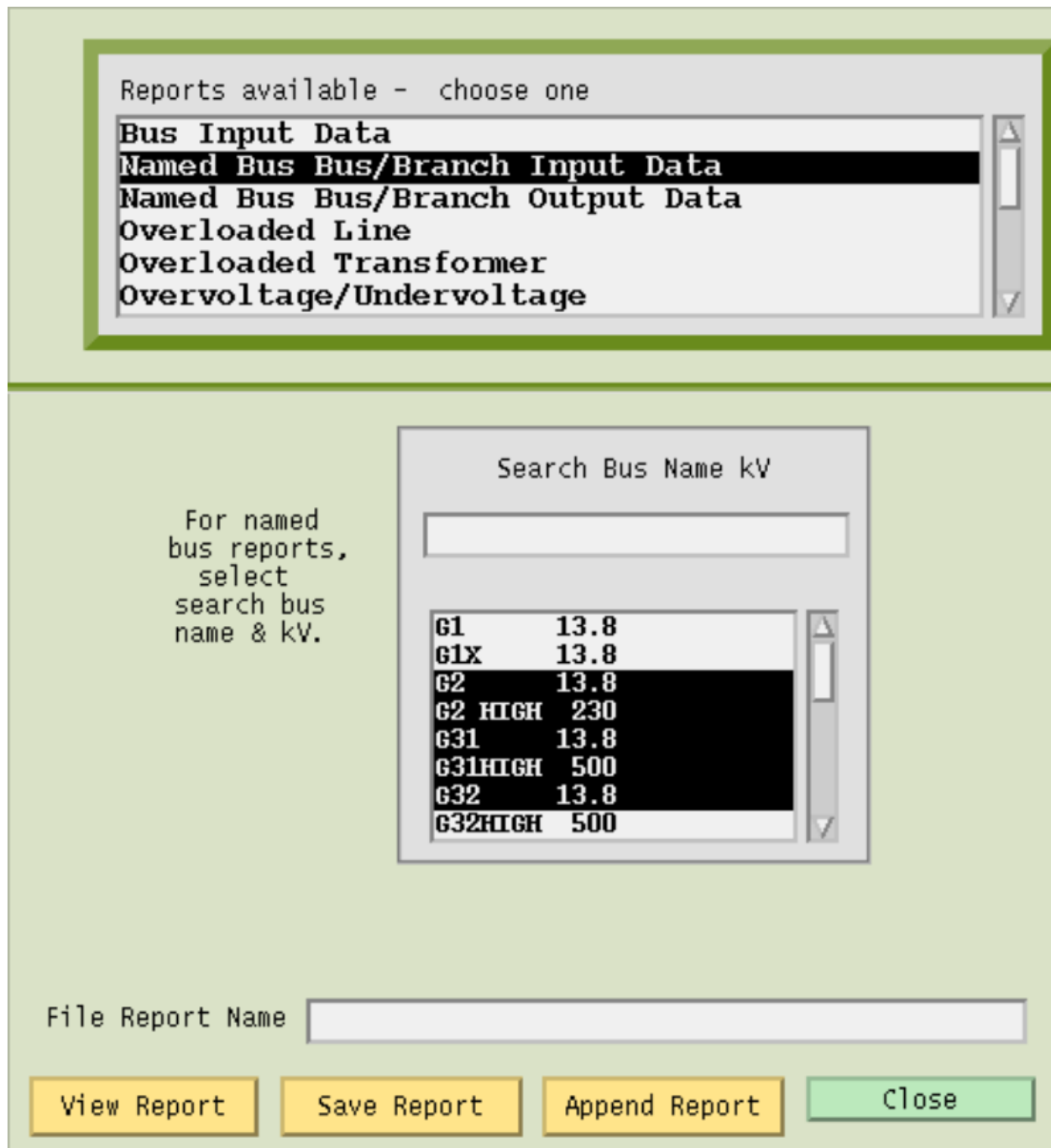


Fig. 2.10.49: Reports Dialog Box: Branch Input Data

Reports available - choose one

- Bus Input Data
- Named Bus Bus/Branch Input Data
- Named Bus Bus/Branch Output Data
- Overloaded Line**
- Overloaded Transformer
- Overvoltage/Undervoltage

Area	Zone	Owner	Base kV	Type	Limits
AREA 1	01		13.8	B	% Line Load <input type="text"/>
AREA 2	02	BPA	22.0	BC	
AREA 3	03	LAP	113.0	BD	
			230.0	BE	
			500.0	BF	
				BG	

File Report Name

Fig. 2.10.50: Reports Dialog Box: Overloaded Lines

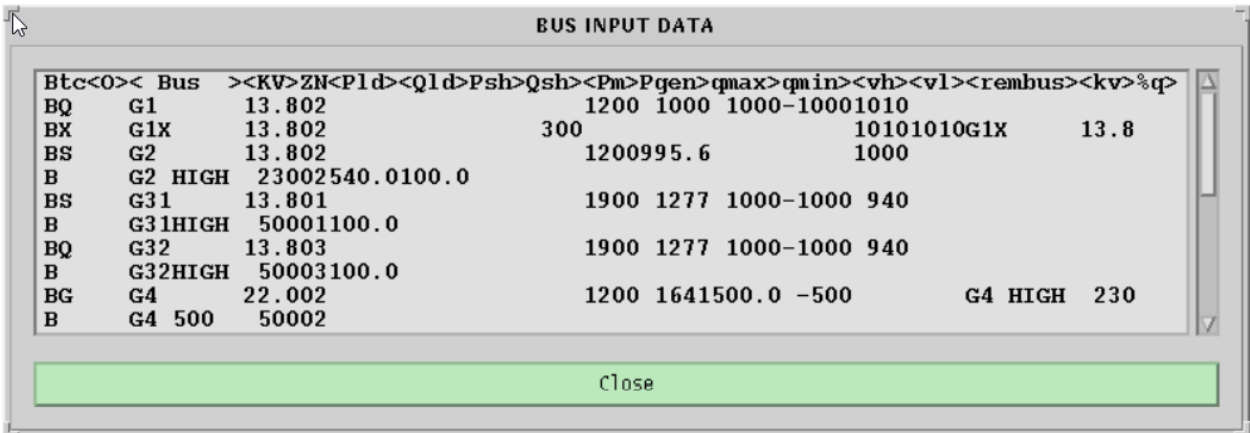


Fig. 2.10.51: Bus Input Data Report Example



Fig. 2.10.52: Bus/Branch Input Example

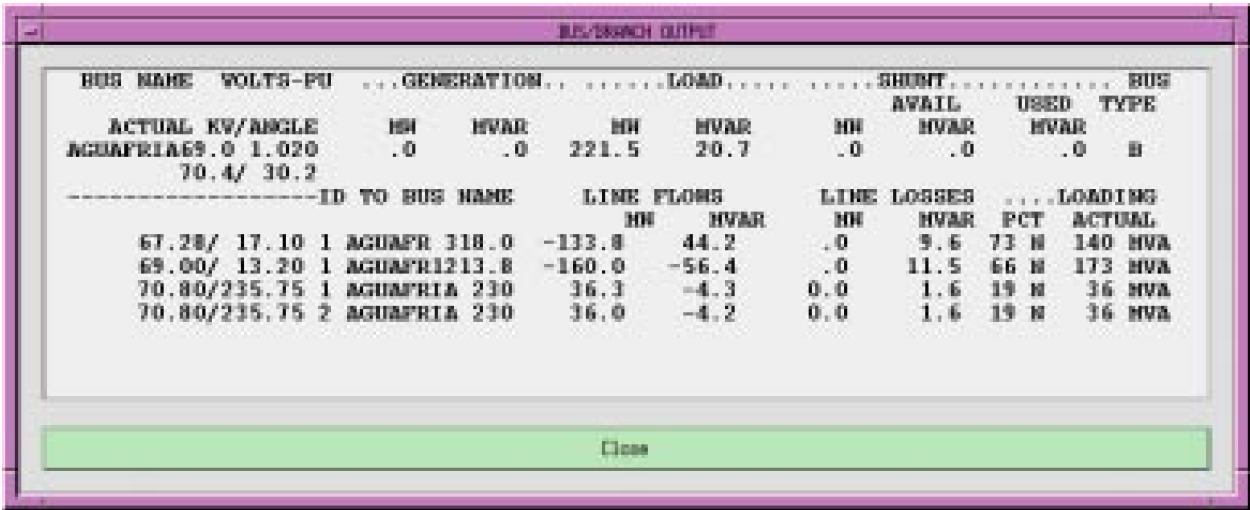


Fig. 2.10.53: Bus/Branch Output Example

Area Interchange Summary							
Area interchange summary case 0102HW1 date 5/20/94							
Area name	Slack bus	Slack bus generation					
	Name	Base	Beginning MW	Change MW	Final MW	Maximum MW	Net Interchange MW
ARIZONA	AGUAFR 3	18.0	131.7	2.1	133.8	181.0	4685.4
B.C. HYDRO	BRIDGE R	13.8	195.8	9.6	205.4	514.0	1823.5
IDAHO	BROWNLEE	13.8	297.8	.1	297.9	450.0	1409.0
IMPERIALCA	ELSTEAMP	92.0	211.6	-.1	211.5	265.0	352.2
LADWP	HAYMES5G	18.0	235.8	-8.1	227.7	341.0	-1935.0
MEXICO-CFE	TTD1	13.8	72.7	0.0	72.7	160.0	220.0
MONTANA	KERR	13.8	99.6	-.4	99.2	180.0	1064.0
NEVADA	SUNRS1GN	13.8	47.0	-.1	46.9	90.0	-1288.0
NEW MEXICO	MEMPHANG3	13.8	46.6	0.0	46.6	103.0	-269.5
NORTHWEST	COULEE 2	13.8	2127.0	-10.1	2116.9	2615.0	-265.0
PECO	HAUGT G1	18.0	136.8	-1.8	135.0	167.0	-435.6
PG AND E	PTS7SHNG	20.0	592.4	.6	593.0	710.0	-980.0
PSCOLORADO	CHEROKEE	20.0	133.6	0.0	133.6	150.0	-902.7
SANDIEGO	ENCINA 4	22.0	138.6	.3	138.9	292.0	-1417.4
SIERRA	TRACY G3	13.8	32.6	.1	32.7	115.0	-123.0
SOCALIF	ALAMT5 G	20.0	412.2	1.2	413.4	480.0	-4716.0
TRANSMIA	BRAZEAU	13.8	153.7	-.1	153.6	330.0	.0
W KOOTENAY	HANETA A	14.4	179.6	-.1	179.5	180.0	-323.5
WAPA L.C.	HOOVERA3	16.5	97.2	.5	97.6	130.0	1409.9
WAPA L.H.	HIPP-1	24.0	453.1	-.3	452.8	550.0	-383.3
WAPA U.C.	GLENCAH6	13.8	617.2	.4	617.6	809.0	2103.0
WAPA U.H.	PT PECK1	13.8	51.0	-.5	50.5	61.8	-28.0

Fig. 2.10.54: Area Interchange Report Example

TIE LINE SUMMARY							
Tie line summary of area interchange case 0102HW1 date 5/20/94							
Area 1	Area 2	/----- Intertie branch -----/					
		Zone	Bus 1		Zone	Bus 2	line flow
01DAMO	LADWP	IP	DRY LAKE	500.0	13 *	MARKETPL	500.0 84.8
		IP *	ROBINSON	500.0	13	DELTA	500.0 -128.8
		IP	HELLSCYN	230.0	NH *	ENTERPRS	230.0 194.5
0	NORTHWEST	IP	HINES	138.0	NF *	HARNEY	115.0 7.2
		IP	LADD	230.0	NH *	LACRANDE	230.0 25.8
		IP *	MIDPOINT	500.0	NF	BURNS	500.0 640.5
0	PEOE	IP	OXBOW	230.0	77 *	LOLO	230.0 -74.2
		IP *	AMFLS	138.0	L1	NALAD	138.0 13.6
		IP	BLACKFT	161.0	L1 *	GOSHEN	161.0 -62.4
		IP *	BORAH	345.0	L1	BENLOMB	345.0 -27.2
		IP	BRADY	230.0	L1 *	ANTLOPE	230.0 -14.7
		IP *	BRADY	230.0	L1	TREASTON	230.0 19.1
		IP *	BRIDGER	345.0	L1	BRIDGER	230.0 204.3
		IP	BRIDGER	345.0	L1 *	GOSHEN	345.0 550.7
		IP *	KIMFORT	345.0	L1	GOSHEN	345.0 -110.8
		IP *	MIDPOINT	345.0	27	IDAMO-NV	345.0 6.6
Area total				(MW)		(MVAR)	
		Generation		3834.9		443.0	
		Load		2275.0		987.0	
		Losses		150.9		-289.7	
		Shunt		.0		-207.4	
		Net Export		1409.0		-461.8	
Close							

Fig. 2.10.55: Tie Line Summary Report Example

## Run CFLOW (Process)

CFLOW files are C programs using the IPF CFLOW library of routines. These routines enable you to access the base case memory-resident data in IPF. Many CFLOW programs are designed to collect information for specialized reports that are not built into IPF in the Reports command. See *CFLOW C API (libcflow)* for more information.

There are two ways to execute a CFLOW program from the GUI. Refer also to *Auto CFLOW* earlier in this section so that you choose the appropriate CFLOW execution command.

The Run CFLOW command is used primarily during the development and debugging of CFLOW programs. During this time, you use a terminal window outside of IPF as you work on getting the program logic to function properly.

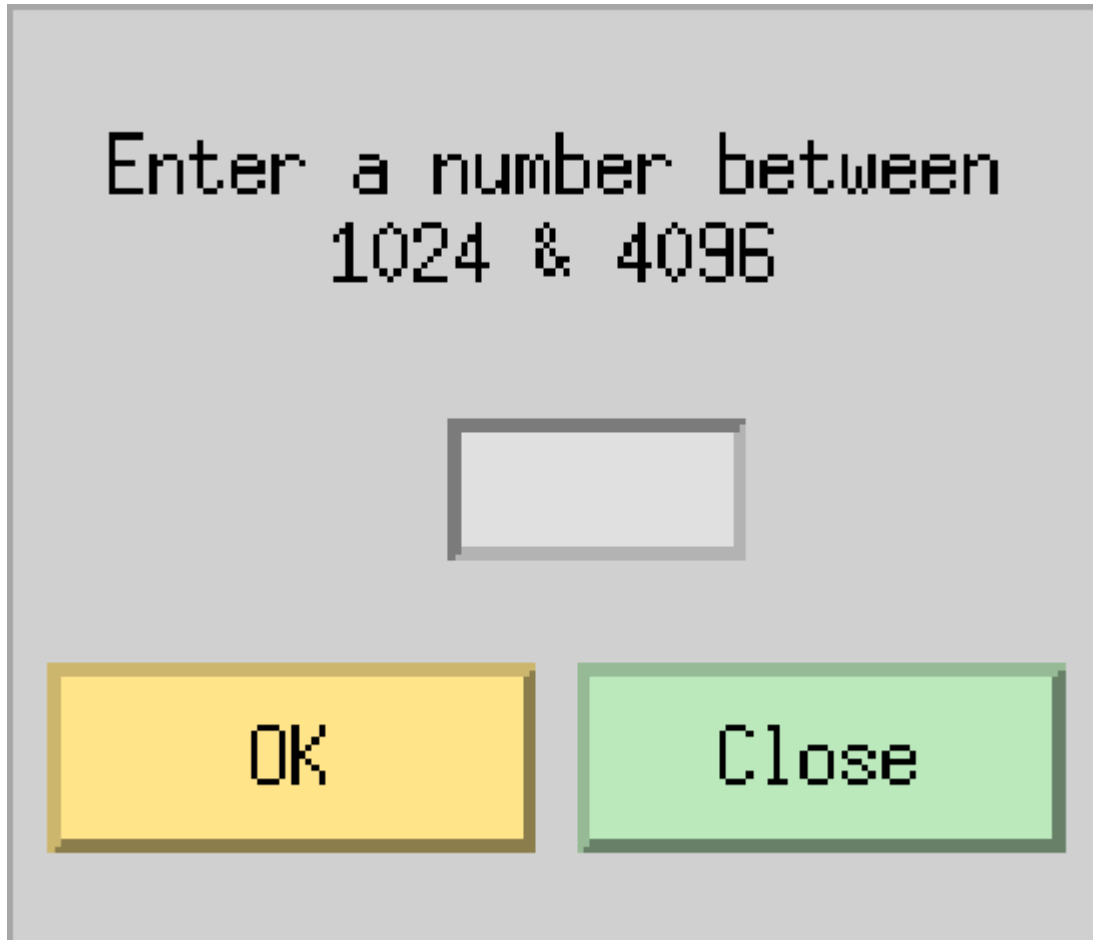


Fig. 2.10.56: Run CFLOW Dialog Box

A second stage of CFLOW programming, after the basic program logic is correct, may involve directing input and output to a separate CFLOW-program-created terminal window. This “production-grade”, user-friendly CFLOW program may then be run by a user from within IPF with the Auto CFLOW command.

### To use the Run CFLOW command:

1. In a terminal window (or other programming environment) outside of IPF, develop and debug the basic programming logic of your CFLOW program.
2. When you are ready to test it with IPF, start up IPF in a *different* terminal window.
3. Start your CFLOW program in its terminal window with a socket id number between 1024 and 4096 on the command line. Example: `cfdgm 2020`
4. Return to IPF, select the Run CFLOW command and type the same socket id number that you typed on the command line in the terminal window.
5. Click OK to supply the socket id number to your CLFOW program and start it executing.

## Save (File)

The Save command permanently stores current work on a disk storage device. When you are through running a solution on a network or editing the coordinates in a network, you usually want to keep the work. You use Save to do this. The saved files remain available for continued modification within the current IPF session. You can save files at any time. See figures below.

The Save dialog box is divided into five file save text boxes with associated Save buttons, and Options buttons for the Network and Stability files. The current files that you have loaded appear in their respective text boxes when you first open this dialog box. If you want to change the name of a file, modify the file name by selecting and typing new characters. You may also add a file path if you want to save the file in a different directory.

If you fail to change an existing file name to a new name, you get a dialog box asking you to confirm that you really want to overwrite an existing file.

When you save a file, IPF does not change the file names in the Open dialog box or any file names in the Current Files Area of the main window. These file names change only if you use the Open dialog box to load new files.

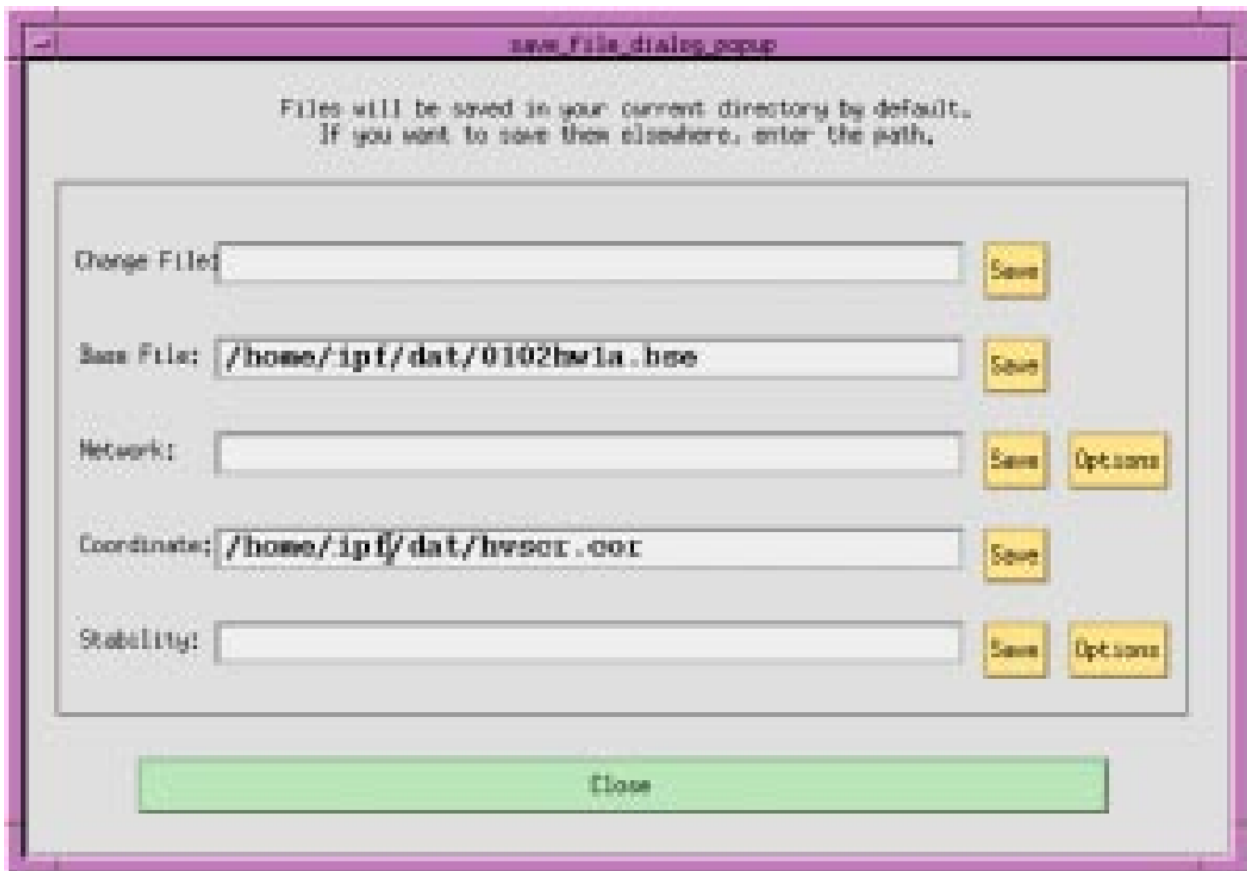


Fig. 2.10.57: Save Dialog Box

**Change File.** This ASCII text file is made up of data representing modifications made in the current session.

**Base File.** This is the main, binary format output file of IPF after it has run a solution.

**Network.** This ASCII text file contains the bus and branch record data comprising the current network, including any alterations you have made.

**Coordinate.** This ASCII text file contains position and identification data for buses and branches in the current case.



**Stability.** This ASCII text or binary file contains base case data that is readable by the WSCC Stability program.

**Save.** These Save buttons are dedicated to their respective files. Click the appropriate button(s) to save the file(s) you want to keep.

**Options.** These Options buttons are dedicated to their respective files. Click the associated button to pop up a dialog box appropriate to the Network or Stability file.

**Close.** Clicking this button closes the dialog box and returns you to the current IPF session.

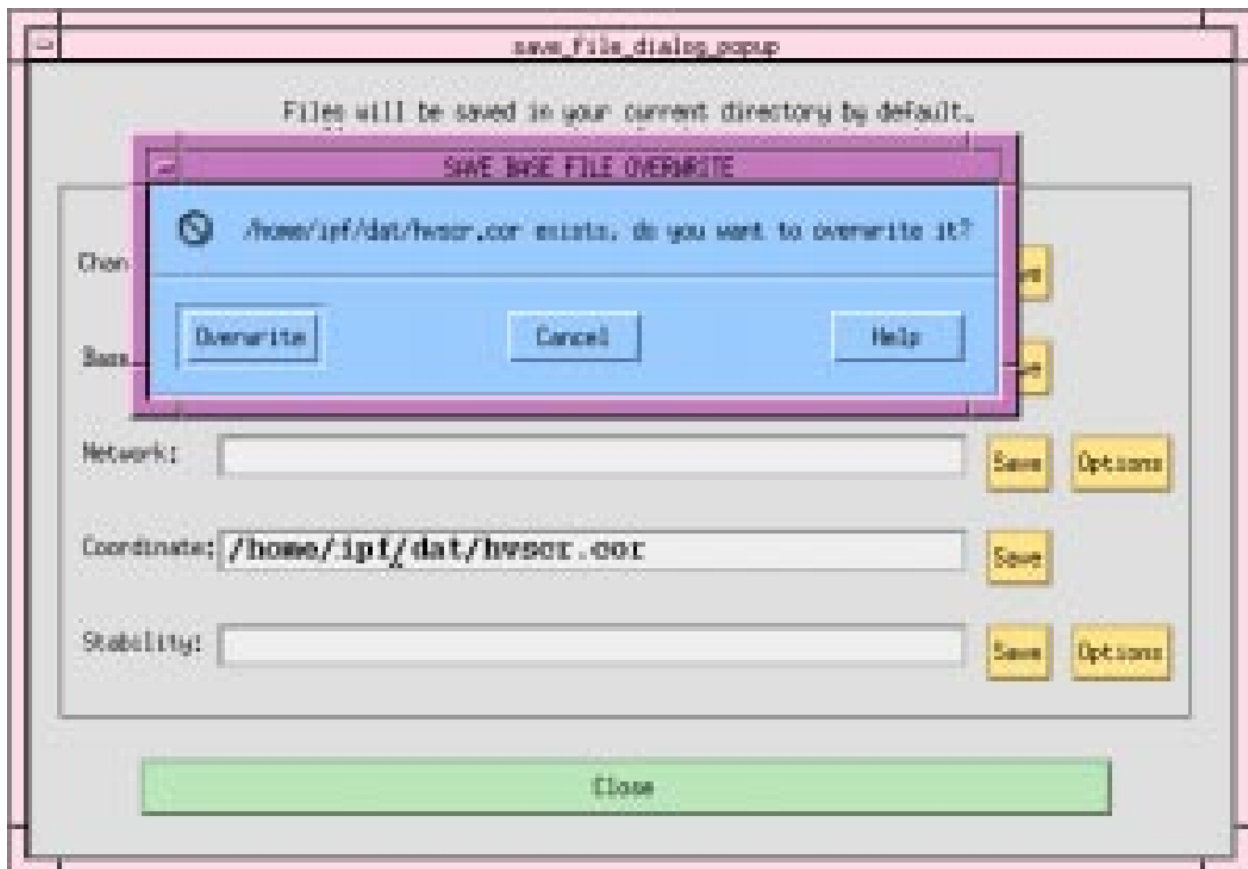


Fig. 2.10.58: Save Overwrite Dialog Box

**Overwrite.** Clicking this button causes the data in the file named in the dialog box to be overwritten. This is a replacement operation, and, thus, you may lose data if you are mistaken. Caution is urged.

**Cancel.** Clicking this button closes the dialog box and returns you to the main Save dialog box.

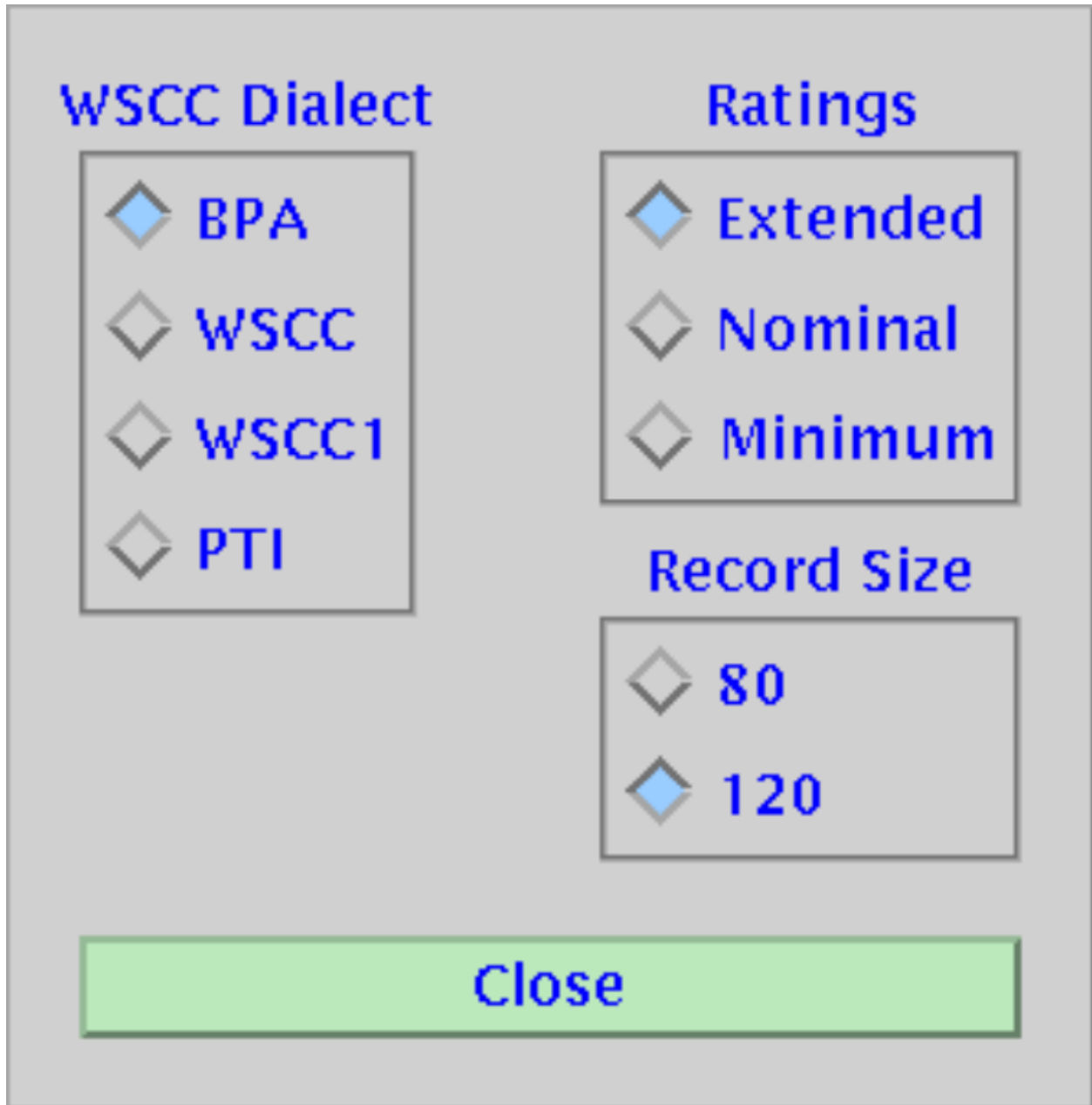
**Help.** (Not yet implemented.)

**WSCC Dialect.** Dialect refers to how values in a field are interpreted and how fields are arranged. See *ipfcut* and *ipfnet* for more information. BPA is the default.

**Ratings.** These refer to line voltage ratings. Extended means that more than one rating is allowed for a line. For example, thermal is another kind of rating. Nominal means there is just one rating allowed. Minimum means that the lowest of extended ratings is selected for the file. Nominal is the default.

**Record Size.** Eighty (80) specifies that you want to limit the record size to 80 characters. WSCC, WSCC1, and PTI dialects require this. The BPA dialect allows the 120 setting.

**Close.** Clicking this button closes the dialog box without writing the file to disk or changing any radio button values.



The dialog box is titled "Save Network Options" and contains three main sections: "WSCC Dialect", "Ratings", and "Record Size". Each section has a list of options, each preceded by a diamond-shaped radio button. The "WSCC Dialect" section has four options: BPA, WSCC, WSCC1, and PTI. The "Ratings" section has three options: Extended, Nominal, and Minimum. The "Record Size" section has two options: 80 and 120. A green "Close" button is located at the bottom of the dialog box.

WSCC Dialect	Ratings	Record Size
<input checked="" type="radio"/> BPA	<input checked="" type="radio"/> Extended	<input type="radio"/> 80
<input type="radio"/> WSCC	<input type="radio"/> Nominal	<input checked="" type="radio"/> 120
<input type="radio"/> WSCC1	<input type="radio"/> Minimum	
<input type="radio"/> PTI		

Close

Fig. 2.10.59: Save Network Options Dialog Box

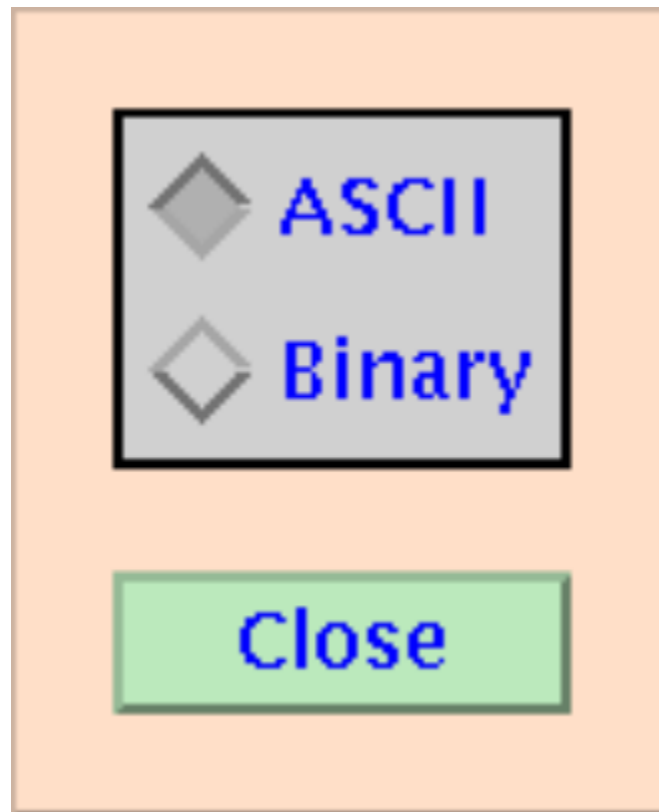


Fig. 2.10.60: Save Stability Options Dialog Box

**ASCII, Binary.** The ASCII button saves the solved system data in ASCII text format, which is readable by ordinary text editors. This file can be transferred to any other platform for input to the WSCC stability program. The Binary button saves the same data in binary format. The binary file is less than half the size of the ASCII file, but can be used as stability input only on the same type of computer system.

**Close.** Clicking this button closes the dialog box without writing the file to disk or changing any radio button values.

### Solution Data Off (View)

The Solution Data Off command rewrites the display *without* reading and displaying the current solution data. Sometimes you may want to clear the display of extra data so that it is more readable. This command does that.

### Solution Data On (View)

The Solution Data On command rewrites solution data to the display. When you do repeated solutions, the displayed solution data does not automatically refresh in all circumstances. When you notice this, do a Solution Data On command to read and display the latest solution data.

### Solve Case (Process)

Solving a network case (or base case) causes IPF to calculate bus voltages that satisfy network constraints as they exist within the currently resident system data. See below. In a solution scenario, the following steps are typical:

1. You make changes to the case.
2. You solve the network.
3. You examine the output.

You follow these steps repeatedly until some desired output conditions or criteria are achieved. The Error Message dialog box gives you feedback about the progress and success of a solution attempt. See below. However, the iteration report is visible only in the terminal window behind the GUI.

Since solution voltages are stored in all base cases, you need not solve a case after initially loading one to access solution voltage data. However, if your case is specified by a network data file, you do need to run a solution to access solution voltage data. Thereafter, solution voltages are stored in the binary base case data.

You do not need to load a coordinate or change file to solve a case — only a base case file or network file is needed.

**LTC.** The radio buttons in this section select different load tap changing (LTC) transformer options. Button On enables all LTC transformer options. RP & RQ enables only LTC transformers controlling real and imaginary power. R Only enables only LTC transformers controlling voltage. DC Only enables only LTC transformers controlling DC converters. Off turns all LTC transformers off.

**Area Interchange.** The CON radio button causes IPF to control the area interchange according to the specified constraints. MON does not control, but will inform you of constraint violations. The OFF radio button causes IPF to ignore all area interchange constraints.

**Phase Shifter Bias.** There are two options for phase shifter bias: BPA and WSCC. Choose BPA to instruct IPF to try to bias voltage phase angle to zero if possible for minimum losses. Choose WSCC to instruct IPF to try to bias voltage phase angle to the initial values of the base case.

**Limits.** QRes is the per unit MVAR by which a bus must be perturbed to revert from a state of Qmax control to a state of V control. Min Phase Angle is the minimum angle in degrees for which fixed-tap phase shifters are modeled as ideal devices in the DC iterations. Delta Angle is the maximum angle adjustment in radians permitted in one iteration. Delta Volts is the maximum voltage adjustment in per unit permitted in an iteration.

**Tolerances.** These values set the tolerances in per unit for convergence testing.

**SOLVE**

**LTC**

☒ On  
☐ RP & PQ  
☐ R Only  
☐ DC Only  
☐ Off

**Area Interchange Control**

☒ CON  
☐ NON  
☐ OFF

**Phase Shifter Bias**

☒ BPA  
☐ MSCE

**Limits**

OPR:   
 Min Phase Angle:   
 Delta Angle:   
 Delta Volts:

**Tolerances**

BUSV:   
 AT POWER:   
 TX:   
 Q:

**Iteration Control**

Decoupled Start:   
 Newton-Raphson:   
☐ Iteration Summary

☐ Base Solution

**Debug**

☐ TX  
☐ BUS  
☐ AL  
☐ DC

**Misc Control**

☒ Flat Start  
☒ DCLP

**BK Voltage Bias**

☒ BPA  
☐ MSCE  
☐ WPMX

Tap Start:   
 VSTEPS:

**Buttons:**

Fig. 2.10.61: Solve Dialog Box

**Iteration Control.** This section specifies the number of iterations IPF goes through. Specifying a number for Decoupled Start means that you want IPF to calculate P and Q separately before it goes to the full Newton-Raphson solution method. Values from zero (0) to ten (10) are valid. Specifying zero means that you want to skip the decoupled start. Specifying two means that one P solution and one Q solution is done. Specifying ten means that five P and five Q solutions are done, alternating P and Q (P, Q, P, Q, P, ...) before the Newton-Raphson solution begins. It is often useful to do more DC iterations if a case has early difficulty in solving.

If a case is 'hunting', you can increase the number of Newton-Raphson iterations beyond the default of 30 to cause it to keep trying. Setting it lower (say to 10) will not cause it to stop before it finds a solution, however. Maximum is 50.

Turning the Iteration Summary button on causes several thousand lines of solution process detail to be written to your [logon].pfd file.

**Base Solution.** Clicking this button bypasses the solution routine, and uses the base voltages in residence to calculate line flows.

**Debug.** These buttons turn on various program debug switches. See the ?? for more details.

**Miscellaneous Control.** If the Flat Start radio button is on, IPF will initialize all bus voltage angle values to zero before it does a solution. This is the default. If the button is off, IPF will leave all bus voltage angles and magnitudes unchanged from their previous solution values. This option is chosen most often for base cases that have already been solved, and to which only minor changes have been made. The DCLP button should be on. It controls the type of solution used for multi-terminal DC lines.

**BX Voltage Bias.** There are three options for BX Voltage Bias: BPA, WSCC, and VMAX. The BPA option accepts any discrete reactance step on a BX bus when its solution voltage  $V$  lies between  $V_{min}$  and  $V_{max}$ . The VMAX option attempts to find the switched reactance step such that the solution voltage is the largest which is still less than or equal to  $V_{max}$ . The WSCC option adjusts the shunt only when the voltage violates the limits.

**Tap Start.** Sets the LTC transformer starting tap. See ?? for a full description of how the starting tap is calculated.

**VSTEPS.** This controls the modeling of BQ buses which are on a Q limit.

**Reset.** Clicking Reset returns all radio buttons to their default values.

**Solve.** Clicking the Solve button causes the IPF solution algorithms to attempt a solution within the constraints of the base case data and the Solve dialog box option settings. Solution performance varies with computer system and size of the base case data set. Messages about the solution can be seen in the Error Messages dialog box. See below. If there are any Fatal errors, this box will pop up on its own; otherwise you must open it in order to see the messages. The iteration history can be observed in the terminal window where IPF was started. You can examine all of this in the [logon].pfd file in your current directory, after you exit IPF.

**Close.** Clicking Close closes the dialog box without saving any changes in the settings.

### 2.10.31 Customizing the GUI (XGUI)

X resources are X system components managed in common by the X server. Examples of resources are colors, fonts and their characteristics, default size and position of windows, and default file names for dialog boxes. You can change many of these resource values in the IPF gui.

You customize IPF gui, as you do other X clients, by changing the client's X resources file. This is the XGUI file in your home (logon) directory. It is an ASCII text file, so you can alter it with any text editor. It is recommended that only advanced users edit XGUI to any extent; when making changes to X resource values it is very easy to cause problems that you won't know how to fix. For example, if you accidentally changed some branch colors to the main window's background display color, you will not be able see those branches, even though they are still there! Caution is urged when you modify X resource values and specifications.

However, there are a few items that anybody can and will probably want to change. These are covered below.

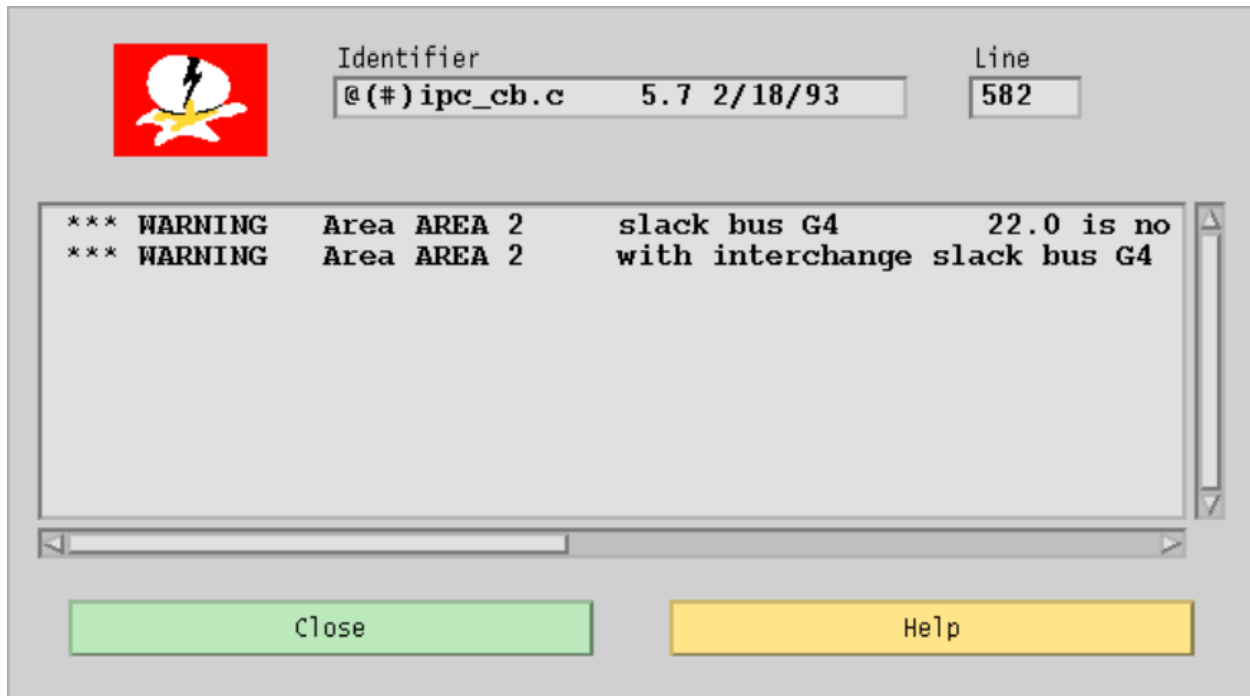


Fig. 2.10.62: Solution Feedback

**Note:** After you make changes to the XGUI file, you may not see changes in your client (IPF) till you exit the X Window System itself. Exiting and restarting IPF is not sufficient! You must exit IPF and the X Window System and restart both.

## XGUI Resources

As you look at the following XGUI file excerpts, note these characteristics:

- For the most part, each line specifies a resource and value for the resource.
- Each resource specification consists of a descriptive resource name followed by a colon (:) followed by a value.

- Values may be numeric or alpha.
- Comment lines are preceded by an exclamation point (!).
- Categories of resources are grouped for reading convenience.
- The IPF resources you can change — the ones in the XGUI file — are window size and position, fonts, colors, and file names.

### Changing IPF Resources

Most values for resources are easy to figure out. Others, like colors, are more difficult. Here are a few hints to get you started. Be sure to read further in Quercia's book, or a comparable book, for more information.

**Warning:** If you make a spelling mistake or the value is not correct for the resource, there are no error messages. The resource is ignored. And, there are no X Window System facilities to help you find your error, so be careful!

### Changing Open File Defaults

Although you can always find your way to the file you want to open by using the Files filter system, it is much nicer to have IPF come up with the paths and/or names you most commonly use. Below is the section of the XGUI file which you will want to alter in order to do this. Open XGUI with your text editor and forward search for "DEFAULT MASKS"

```
! These are the DEFAULT MASKS in the open file menu.

XGUI*open_dia_command_dir_text.value: *.pcl
XGUI*open_dia_change_dir_text.value: *.chg
XGUI*open_dia_base_dir_text.value: *.bse
XGUI*open_dia_network_dir_text.value: *.net
XGUI*open_dia_coord_dir_text.value: *.cor

! These are the default filenames in the open file menu.
! Leave blank where you don't want a default.
! Don't set default filenames for both base and network!!!

XGUI*file_select_dia_command_text.value:
XGUI*file_select_dia_change_text.value:
XGUI*file_select_dia_base_text.value:
XGUI*file_select_dia_network_text.value:
XGUI*file_select_dia_coord_text.value:
#####
```

When you first execute IPF and select the Open command from the main window menu, some default file selections fill the file text boxes under the Select buttons. These specify the default filters to be applied.

**To change a default filter mask:** For example, the default coordinate file mask in the XGUI file for the Open dialog box is:

```
XGUI*open_dia_coord_dir_text.value: *.cor
```

This assumes that you are executing gui from the directory where the data files reside, and that you will be selecting the coordinate file you want to load from among files in this directory that end with .cor. (The asterisk (\*) is a Unix wildcard character meaning "any arbitrary length string of characters.")

To make the filter default to a different directory, insert (for example) /archive/ipf/dat/:

```
XGUI*open_dia_coord_dir_text.value: /archive/ipf/dat/*.cor
```

Now the coordinate files selected are in the /archive/ipf/dat directory. The pathname you specify can be absolute or relative; just be sure that it specifies a valid directory on your system.

**To change a default filename mask:**



The default base case and coordinate file masks in the XGUI file for the Open dialog box are specified with these two lines:

```
XGUI*file_select_dia_base_text.value:
XGUI*file_select_dia_coord_text.value:
```

The filenames are currently blank, meaning that no files are loaded when gui starts up. If during a study you are working continuously from a particular base file, say `../base/97HS4.bse`, you might want to edit XGUI to make this file, and its associated coordinate file, load automatically every time you start gui. You then can begin work immediately, without having to go through the Open Files menu.

To add default filenames, insert them after the :

```
XGUI*file_select_dia_base_text.value: ../base/97HS4.bse
XGUI*file_select_dia_coord_text.value: /archive/ipf/dat/volts.cor
```

These two files will be automatically loaded for you. If you look at the Open Files menu, you will see their names in the boxes to the right of the filter boxes.

**Warning:** Do not specify both a base file and a network file name. These are mutually exclusive!

## Changing Printer Defaults

You can specify your default printer and also enter a list of others you may want to choose from. These are right at the top of the XGUI file.

```
XGUI*printer_selection_box*textstring: print
```

Change “print” to whatever string you will want to use most often.

The section just below this is where you put in the other print strings that you want to appear in the printer list.

## Changing Window Position and Size

Windows are positioned on the screen by specifying pixel locations. X is the horizontal dimension. Y is the vertical dimension. The position  $x, y = (0, 0)$  is the upper left corner of the screen. `(XGUI.x, XGUI.y)` represents the upper left-hand corner of the main window of IPF.

### To change the main window default position:

The default in the XGUI file is::

```
XGUI.x: 127
XGUI.y: 0
```

Change the 0 to a 127::

```
XGUI.x: 127
XGUI.y: 127
```

Now the upper left-hand corner of the main window appears 127 pixels down and to the right of the 0,0 screen pixel position in the extreme upper left-hand corner of the screen.

### To change the main window default size:

The default in the XGUI file is::

```
XGUI.width: 600
XGUI.height: 550
```

Change the 550 to a 600::

```
XGUI.width: 600
XGUI.height: 600
```

Now the IPF main window will be square. Other windows and dialog boxes in XGUI work similarly.

### Changing Fonts

The X Window System for your computer comes with a standard set of fonts. Commonly used fonts such as Courier (a typewriter-like font), Times (a serif, “newspaper” font), and Helvetica (a common, sans serif font) are included. Ask your system administrator about the X fonts available on your computer. Each size, shape, and kind of font has a unique name. Here is the name of one of IPF’s default fonts::

```
-*-Courier-Bold-R-*--*-100-*--*-*-ISO8859-1
```

This X font name specifies Courier bold with a normal slant (R) of size 10 points. Its International Standards Organization character set registry identification is ISO8851-1. The asterisks denote “don’t care” states for the other font parameters such as, for example, foundry (who made the font) and pixel size.

#### To change a default font:

For example, the default font in the XGUI file for descriptive IPF text in the windows and dialog boxes is::

```
XGUI*XmText.fontList: *-Courier-Bold-R-*--*-100-*--*-*-ISO8859-1
```

Change the Courier to Times:

```
XGUI*XmText.fontList:-Times-Bold-R---100----*-ISO8859-1
```

Now the descriptive text is a 10 point bold Times of a regular slant.

### Changing Colors

X color values can be specified as regular names such as blue, red, yellow, magenta, slate blue, sky blue, navy blue, etc., or as hexadecimal digits. Ask your system administrator for a list of all named standard colors because this is much easier to deal with than figuring out hexadecimal color values. However, here is a quick explanation of the hexadecimal color specification system.

Colors are specified in the RGB (Red-Green-Blue) color system. The RGB hex numbers have 12 digits. The first four stand for the Red component. The middle four stand for the Green component. And the last four stand for the Blue component. (Reading left to right, of course.) ffff hex stands for fully saturated red if it is in the first position. 8000 hex stands for a half way saturated red (called “red4” in X). 0000 hex stands for no red at all, which would be black. The green and blue work analogously. Thus, ffffffffffff hex stands for pure white, and 000000000000 hex stands for pure black. To get the color you want, you need to play with different values of hex numbers in the appropriate Red-Green-Blue positions since the RGB intensities are mixed to render one color.

**To change a default color:** For example, the default color in the XGUI file for the 500 kV branches as shown in the branch color key of the IPF main window is::

```
XGUI*kv_500_label.background: #ffffcccc0000
```

Note the hexadecimal color value specification. This default color is a gold-looking color.

Use a color name specification and change the #ffffcccc0000 to yellow::

```
XGUI*kv_500_label.background: yellow
```

Now the 500kV branches and the branch color key shows up as pure yellow.

## Changing Default File Names

When you first execute IPF and select the Open command from the main window menu, some default file names fill some of the file text boxes. You can specify your own valid file names or file name masks in the XGUI file.

### To change a default file mask:

For example, the default coordinate file mask in the XGUI file for the Open dialog box is::

```
XGUI*open_dia_coord_dir_text.value: /shr5/all/ipf/dat/*.cor
```

The \*.cor selects just the coordinate files in the given directory (if they all end with .cor, of course). (The asterisk ( \* ) is a UNIX wildcard character meaning “any arbitrary length string of characters.”)

Change the /shr5/all to /archive/year1992::

```
XGUI*open_dia_coord_dir_text.value: /archive/year1992/ipf/dat/*.cor
```

Now the coordinate files selected are in the /archive/year1992/ipf/dat directory.

You can change any part of the file name, of course, just so long as the file name is a valid file name for your operating system.

## 2.11 CFLOW C API (libcfLOW)

### 2.11.1 Overview

CLFOW is a C Application Programming Interface (API) for IPF. The library name is libcfLOW. With CFLOW, users can access the bus and branch data within the power flow data structures in a highly flexible way. Once data is retrieved from the IPF “data engine”, it can be manipulated within the user-written C program and either output to a file, screen, or plotter, or sent back to the IPF “data engine” for further processing. A CFLOW program is a totally separate process running concurrently with the “powerflow solution and data server” process and communicates with it via an Inter Process Communication (IPC) protocol that uses sockets.

In order to use CFLOW, you must have some knowledge of the C language and have a C compiler. Once a CFLOW program has been written, compiled, linked, and debugged, it is stored as an executable which can be run from the command line or from the IPF GUI using the “PROCESS – CFLOW” menu option.

CFLOW is completely compatible with ANSI C and portable to all environments that have an ANSI C compiler.

### History

CFLOW was written to perform analogous functions to the WSCC's Computationally Oriented Programming Environment (COPE) language. COPE was a standalone language integrated with the WSCC equivalent of BPA's IPF, the Interactive Powerflow System (IPS). CFLOW is a library of C language functions, that effectively wrap the `ipsrv` server component of IPF. This means it operates as a "remote procedure call" library. All the power of the C language is available to CFLOW users, whereas COPE users are limited to the COPE language and environment. Another example of similar functionality is PTI's IPLAN which is interpreted similar to the way COPE is. CFLOW is more powerful and flexible than COPE.

### Audience

This documentation assumes that you are a beginning to mid-level C language programmer. This means you should have successfully written programs in some languages. If you have some experience with Fortran, BASIC, Pascal, or C it will be even easier for you to grasp. You need not have written complex programs, simply programs such as you would be required to complete in an undergraduate college programming course. If you already know C, you are ready to write CFLOW programs. If not, you should probably take a class, invest in a computer tutorial course, and/or spend some time with a good C language book.

The following C programming books are recommended for those users needing introductory or refresher information:

- Brakakati, Nabajyoti. *The Waite Group's Microsoft C Bible*. Howard W. Sams & Company, 1988. This MS-DOS environment reference book clearly describes ANSI C compatibility for each function.
- Harbison, Samuel P. and Guy L. Steele. *C: A Reference Manual*. 3rd ed. Prentice-Hall, 1991. This book shows ANSI C facilities contrasted with traditional or alternate facilities. If you are well acquainted with C programming, but want to make sure your program complies with ANSI C, look here.
- Johnsonbaugh, Richard and Martin Kalin. *Applications Programming in ANSI C*. MacMillan, 1990. This is a textbook used in beginning undergraduate college courses.
- Kernighan, Brian W. and Dennis M. Ritchie. *The C Programming Language*. Second Edition. Prentice-Hall, 1988. This is the standard book for learning the language, updated to a second edition. Experienced programmers will do well with this book. Beginners should use Kochan's book.
- Kernighan, Brian W. and Rob Pike. *The UNIX Programming Environment*. Prentice-Hall, 1984. This book describes how to develop programs in the UNIX operating system.
- Kochan, Stephen G. *Programming in ANSI C*. Howard W. Sams & Company, 1988. This book is a comprehensive tutorial for the beginning programmer.
- Plauger, P. J. *The Standard C Library*. Prentice-Hall, 1992. This book shows you how to use the standard ANSI and ISO C library functions. It provides code examples for implementing many of the library functions.

### 2.11.2 Creating a Program

CFLOW is a library of functions that a C program can link to (with an object file linker) to access IPF data and control IPF execution. The program you write is a C program. CFLOW source programs are created using a text editor. The program lines can be entered in "free format," since there are no column restrictions like Fortran.

Indentation is recommended when designing nested logical constructs to reduce logic errors and enhance readability. For example:

```
if(condition) { /* beginning of first "if" block */
    if(condition) { /* beginning of second "if" block */
        statement;
    }
}
```

(continues on next page)

(continued from previous page)

```

    statement;
}/* end of second "if" block */
statement;
statement;
}/* end of first "if" block */
statement;

```

Include the CFLOW header file, called `cflowlib.h`, in each file that calls a CFLOW function.

### 2.11.3 Running a CFLOW Program

#### From the command line

To run your CFLOW program from the command line, simply run the program as you would any other program from a terminal/command prompt. For example, the following could be used:

```
my_cflow_program [ arg1 ] [ arg2 ] ...
```

The library will take care of launching the `ipfsrv` program and establishing a connection. There are several optional command line arguments that are assumed by every program that uses the `libcflow` library. These options must precede any of the arguments for the CFLOW program.:

- n noserver, do not launch `ipfsrv`
- w <wait time> max wait time for socket connect (default of 30)
- c <socket> socket number to use if other than default of

#### From the GUI

There are two ways to execute a CFLOW program from the GUI: Either in its own terminal window, or in the same terminal window that the GUI is running in (background). CFLOW programs that are run in the same terminal window as the GUI that read/write to `stdin` or `stdout` (i.e. use `readln` or `printf`) have their I/O intermixed in the same terminal window as the GUI and IPF server. The background mode should usually only be used for programs that generate a report or other output to a file.

To run a CFLOW program from the GUI, you start up the GUI, and set up whatever conditions are required for your program to work, such as loading and solving a case, if the program is designed to report on a currently loaded case. Then select Process - Run CFLOW. You will get a file selection window. Double-click on a directory name, or change the filter field and click the Filter button, to change the file list. When you see the file you want to run (your executable CFLOW program), select it, and it will appear in the Selection field. Select either Window (default) or Background, and, if your program has command line arguments, type these into the CFLOW Program Arguments field. Then click the Launch CFLOW button. The Wait field (default 30 seconds) is provided just in case your program takes longer than 30 seconds to start up. The wait time is the length of time that the GUI/`ipfsrv` will wait for the CFLOW program to start up and establish a socket (inter-process communication) connection. The value can be set from 15 to 300 seconds. If the CFLOW program has not connected within the wait time, then the CFLOW run is aborted and control is returned to the GUI.

The program is run synchronously. This means that you cannot use the GUI until the CFLOW program is finished running. If the CFLOW program fails, control is returned to the GUI. However, if the CFLOW program hangs (as in an infinite loop), you need to kill the CFLOW process through operating system resources. (For example, on UNIX this can be done with the `kill` command and on VMS this can be done with the `stop process` command.) See your computer system documentation or your system administrator for help.

When the program completes, the CFLOW window goes away and control returns to the GUI. You can then load a different file, or make other changes, and select Process - Run CFLOW again to rerun. The scripts `run_cflow_win` (window) and `run_cflow_bg` (background) are used to run the CFLOW program. These scripts can be customized for your system.

### From ipfbat

For batch, also called background or terminal window interactive processing, the `ipfbat` program is provided. This program reads a control file rather than connecting to and receiving commands from the GUI process.

A CFLOW program can be run by including the following command in the control file:

```
/CFLOW, PROGRAM =  
[ directory path ] < CFLOW executable file or script file >  
[ , WINDOW ]  
[ , WAIT = < max wait time for socket connect > ]  
[ , ARGS = < command line arguments for the specified program > ]
```

The brackets ( [ ] ) denote optional items. The command is free format with the restriction that any “word” (like the [ path ] < file > ) must be all on the same line (not continued on the next line) with no imbedded blanks or any of the following: “=”, “n”. On a UNIX system, for example, you can use the following:

```
/CFLOW, PROGRAM = my_cflow_program
```

This “launches” the program if it is in your directory search path. `ARGS` is required only if the CFLOW program requires command line arguments. Without the `WINDOW` option, any I/O from the CFLOW program goes to standard input or standard output, and will be to and from the same terminal window that the `ipfbat` program is run from (intermixed with any I/O from the `ipfbat` program). The scripts `run_cflow_win` (window) and `run_cflow_bg` (background) are used to run the CFLOW program. These scripts can be customized for your system.

### 2.11.4 Debugging

Use the standard system debugger for your computer system to debug CFLOW C programs. Although it is possible to do debugging using the GUI/`ipfsrv` or the `ipfbat` programs, it is recommended that you debug by running your CFLOW program from the command line.

If your program expects some setup (i.e. a case already loaded and solved), because your program is used like a subroutine that generates a report or other output based on whatever is currently there, then for debugging purposes, you can create a function that does the setup (e.g. load base, apply changes, solve) and call that function at the beginning of your program. When the program is debugged, you can “comment out” the call to the setup function.

The `pf_cflow_init()` function uses the C library function `system()` to launch the `ipfsrv` program with the output redirected to a file. There is a default wait (time out) of 30 seconds for the `ipfsrv` program to establish a socket connection, but this can be increased, if needed, to up to 300 seconds with the `-w` option. See running [From the command line](#) above. In general, when debugging, you want to “step over” (versus “step into”) the `pf_cflow_init()` function, however, if you decide to step through the function, be aware that part of the code has the time out in effect and will cause a “failed connect” to occur if you proceed too leisurely.

If you experience problems with debugging that you suspect are related to the “system” call that launches the `ipfsrv` program, you can use the `-n` (noserver) option as follows to debug from two windows:

In one window run your CFLOW program.:

```
my_cflow_program -n -w 300 [ arg1 ] [ ... ]
```

Wait for the log message using `socket nnnn` (this will happen when the `pf_cflow_init()` function is executed), then in another window run the `ipfsrv` program.:

```
ipfsrv -socket nnnn
```

where “nnnn” is the same as what the CFLOW program stated. The socket connection should happen within a second. You can then debug your program in one window while the `ipfsrv` program runs in the other.

### 2.11.5 Functions Overview

The CFLOW library is a set of functions, written in the C language, that allows access to IPF data. Behind the scenes, the routines communicate with the IPF program via an interprocess communication channel known as a socket (similar to a pipe or stream). The routines are organized, as much as possible, as data access routines, since the powerflow program is playing the role of data store for the power flow model and solution as well as compute “server.”

There are four major classes of functions:

- Simple IPF “command” functions.
- Record-oriented IPF functions.
- Buffer-oriented IPF “command” functions.
- Utility functions and local data translation.

The library currently accesses only the Powerflow program. In the future, a similar approach could be used to provide a CFLOW interface for other programs. The various functions are documented in Chapter 4. All of those that access Powerflow start with `pf_`. All functions return a non-zero integer for an error condition and a zero for successful completion.

#### Simple Command Functions

Examples of simple functions are `pf_area_of_zone`, `pf_del_zone`, `pf_rename_area`, `pf_rename_zone`, and `pf_rename_bus`. These functions perform an operation that requires little or no input data, other than a command, and usually return only a status, or a single piece of data such as `pf_area_of_zone` does.

#### Record Oriented Functions

Examples of record-oriented functions are `pf_rec_bus` and `pf_rec_cbus`. These functions use C language structures to manipulate a record. Both generic and record-type specific structure definitions are provided, so that field names specific to the record type can be used for a little better “self-documenting” code. For example, the variable containing TAP2 for a transformer contains B2 for an E type line, and the minimum phase shift for a type RM regulating phase shifter. This same variable can be accessed by using the names `r.i.branch.tap2`, `r.i.pf_E.b2`, and `r.i.pf_RM.min_phase_shift_deg`. The table below gives complete details. The columns represent the input values that are defined for branch records. In the table, “N/A” means that the data item does not apply to that record type; “-” means the variable name is the same as the generic variable name (left most column). See the [Record Formats](#) section for descriptions of the various branch records.

Generic branch vari- able name and type	E	L	T	TP	R,RN,RQ,RM,RP	RZ	LD	LM
char type[3]	E	L	T	TP	R,RN,RQ,RM,RP	RZ	LD	LM
char owner[4]	.	.	.	.	.	.	.	.
char bus1_name[9]	.	.	.	.	.	.	.	.
float bus1_kv	.	.	.	.	.	.	.	.
int meter	.	.	.	.	var_tap_side	var_tap_side	var_tap_side	.
char bus2_name[9]	.	.	.	.	.	.	.	.
float bus2_kv	.	.	.	.	.	.	.	.
char ckt_id	.	.	.	.	N/A	N/A	.	I_or_R_control
int sec- tion	.	.	.	.	N/A	N/A	.	N/A
float to- tal_rating	.	.	.	.	N/A	N/A	I_rate	.
int num_ckts	.	.	.	.	num_taps	num_taps	rani_type	N/A
float r	.	.	.	.	N/A	N/A	Pc_max	R
float x	.	.	.	.	N/A	N/A	Pc_min	L_mh
float g	g1	.	.	.	N/A	N/A	Xij_max	C_uf
float b	b1	.	.	.	N/A	N/A	Xij_min	P_sched
float tap1	g2	miles	tap1	phase_shift_deg	max_tap	max_phase_shift_deg	V_sched	N/A
float tap2	b2	N/A	tap2	tap2	min_tap	min_phase_shift_deg	miles	miles
float al- pha_N_deg	N/A	N/A	N/A	N/A	N/A	N/A	.	N/A
float gamma_0_ deg	N/A	N/A	N/A	N/A	N/A	N/A	.	N/A
char de- scrip[9]	N/A	de- scrip[9]	N/A	N/A	rmt_bus_name	rmt_bus_name	N/A	N/A
char date_in[4]	.	.	.	.	.	.	N/A	N/A
char date_out[4]	.	.	.	.	.	.	N/A	N/A
float	.	.	.	.	rmt bus kv	rmt bus kv	N/A	.



The functions all use an action code to specify what is to be done with the record, such as D for delete, F2 to retrieve the first branch record associated with two named buses, and O to retrieve solution (output) data for a bus or branch.

## Buffer Oriented Operations

The buffer-oriented operations are all accessed through one function: `pf_cflow_ipc`. This function sends a buffer to Powerflow containing a command, command options, and usually input data. A buffer is returned that contains the results of the request. Any command that is in the IPF Advanced User Manual can be put in the buffer, with records separated by the 'n' (linefeed) character. Much of the data sent and received in the buffers is WSCC-formatted data.

## Utility Functions

Most of the utility functions are provided to form an “abstraction layer” between your program logic and WSCC formatted ascii records. There are functions to translate between the C structures and WSCC ascii records, as well as functions to initialize the C structures.

## 2.11.6 Notes

Below are a few notes, reminders, and definitions relating to the C language and the CFLOW library functions, for your convenience.

### Strings

Many of the function parameters, such as bus names and action codes, are described as strings. In C, the technical definition of a string is as follows: An array of characters, with a null terminator (ASCII character `\0`) in the element following the last valid character.

The library functions, like most C library functions, expect the strings you supply to conform to this definition. The easiest way is to enclose the value you want in double quotes (e.g. "AMBROSIA"). You can also store a value like this when you initially declare the character array.

---

**Note:** When you declare a `char` array, remember to always size it one larger than you need to store your actual string, so there is room for the null character!

---

However, you cannot put this value in an array with an assignment statement (`name = "AMBROSIA"` is illegal). To store a string in a character array, use the standard C `strcpy` or `strncpy` function.

A single character can be placed in one element of a character array by enclosing it in single quotes (`name[0] = 'A';` is a legal C assignment statement). If you do this, be sure to store a null in the last character (`name[8] = '\0';`).

### Arrays

In C, an array dimension is declared to be the actual number of elements in the array:

```
char name[9];    /* Sized to hold an 8-character bus name, plus a null.*/
```

But when you reference the array elements, the indices run from zero to one less than the declared dimension. In the example above, `name[0]` has the first character of the bus name; `name[7]` has the last character, and `name[8]`, which is the last element of the array, contains a null.

### Function Types

If a function is declared as `void`, then no return value is expected, and it may be invoked without a place being provided to receive the returned value. Example:

```
pf_init_bus( &b, "B", "AMBROSIA", 230.0 );
```

Most of the CFLOW functions are `int`, meaning that they return an integer value, which is usually zero for success and non-zero for any kind of error. These must be called in some way which is compatible with the name of the function being a variable with a value. Example:

```
int error;  
error = pf_rec_bus( &b, "G" );
```

### Main

Your main program must have the standard arguments `int` and `char **`:

```
int main ( int argc, char *argv[] )  
{  
    ...  
}
```

You cannot merely call it “main”, as you may have done for programs in a C class, or which you will see as examples in the books. Look at the sample CFLOW programs for guidance.

### Includes

In most program you write, you at least need the standard C headers `stdio.h` and `string.h`, and also the CFLOW library header, `cflowlib.h`. This is done with a preprocessor statement: `#include <stdio.h>` for libraries in the standard C include directory. `#include "cflowlib.h"` for libraries in some other directory.

### Linking

Your CFLOW program can be compiled by itself, but in order to execute, it must be linked with the cflow library, `libcflow.a`. The simplest way to do this (on a Unix system) is to use something like CMake or a Makefile. Look at the examples in this project for how to do this.

### Buffers

CFLOW communicates with ipfsrv by using buffers which are passed back and forth (see the next section, Global Buffers, for detailed information). If you want to see what is in the buffers, set the variable `cf_debug` to one. As long as it has this value, the contents of every buffer passed will be displayed in the terminal window. Since this output will go by so fast you can't read it, and it can be very voluminous, you will want to limit the number of buffers actually displayed to the ones you are interested in. Set `cf_debug` back to zero to turn off the display.

## Languages

CFLOW routines do not have to be written in C, except for the main program. If you are adept at the intricacies of calling C routines from a Fortran program and vice versa, you can write your main processing and reporting routines in Fortran, or use code you already have. In general, it will be easier to write a C program to perform the function, rather than trying to retrofit.

### 2.11.7 Simple Report Example

The ANSI C program listed below was derived from a COPE program. This program illustrates how the CFLOW library and ANSI standard C may accomplish many of the same tasks as COPE programs. Detailed discussion follows the program code. The following program outputs a Shunt Reactive Summary report to the screen. A loaded base case in the Powerflow “data engine” provides the data for the report.

```
/*
 * The following is an example of a COPE program, re-written
 * in the "C" programming language using the CFLOW library.
 * It does a Shunt Reactive Summary report on the currently
 * loaded case, for a user-entered zone. Output to gui T/W.
 */

#include <stdio.h>
#include <string.h>
/* use this form if "cflowlib.h" is in a "user
library" include area. Your compile procedure
should use the "-I or /I" option that specifies
the path to "cflowlib.h" */
#include "cflowlib.h"

int readln( char *s, int lim ) /* Function to read input from the T/W. */
{
    int i;
    char c;
    for ( i=0; i < lim - 1 && ( c = getchar() ) != EOF && c != '\n'; ++i )
s[i] = c;
s[i] = '\0';
return i;
}

main( int argc, char *argv[] ) /* Main Program */
{
    pf_rec r; /* CFLOW structure */
    char zn[3];
    int error, status ;
    float q_avail_react_tot, q_avail_cap_tot,
q_used_react_tot, q_used_cap_tot,
q_unused_react_tot, q_unused_cap_tot,
q_unsched_react_tot, q_unsched_cap_tot;
    pf_cflow_init( argc, argv ); /* IPC connection function, required. */
    /* Ask user for zone to report */
    printf("Enter Zone to report Shunt Reactive Summary > ");
    readln( zn, sizeof(zn) );
    zn[sizeof(zn)] = '\0';
}
```

(continues on next page)

(continued from previous page)

```

printf("\n\n Shunt Reactive Summary for Zone %s \n\n",zn);
printf(" Avail_caps Avail_reac Used_caps Used_reac Unus_caps Unus_rx Unsch_caps
Unsch_rx\n\n");
q_avail_react_tot = q_avail_cap_tot =
q_used_react_tot = q_used_cap_tot =
q_unused_react_tot = q_unused_cap_tot =
q_unsched_react_tot = q_unsched_cap_tot = 0.0;
/* Compute zone quantities */
error = pf_rec_bus( &r, "F" ); /* get first bus in case */
status = pf_rec_bus( &r, "O" ); /* get solution data for first bus */
while ( !error && !status ) { /* Loop through all buses in case */
if ( strcmp(r.i.ACbus.zone, zn )==0)
{ /* If bus is in the zone */
q_avail_react_tot += r.s.ACbus.Bshunt_sch_rx;
q_used_react_tot += r.s.ACbus.Bshunt_used_rx;
q_avail_cap_tot += r.s.ACbus.Bshunt_sch_cap;
q_used_cap_tot += r.s.ACbus.Bshunt_used_cap;
if( r.s.ACbus.Qunsch < 0 ) {
q_unsched_react_tot -= r.s.ACbus.Qunsch;
}
else
{
q_unsched_cap_tot += r.s.ACbus.Qunsch;
}
}
error = pf_rec_bus( &r, "N" ); /* get next bus in case */
status = pf_rec_bus( &r, "O" ); /* get solution data for next bus */
}
q_unused_react_tot = q_avail_react_tot - q_used_react_tot;
q_unused_cap_tot = q_avail_cap_tot - q_used_cap_tot;
/* Print zone summary */
printf(" %6.1f %6.1f %6.1f %6.1f %6.1f %6.1f %6.1f %6.1f \n\n",
q_avail_react_tot, q_avail_cap_tot,
q_used_react_tot, q_used_cap_tot,
q_unused_react_tot, q_unused_cap_tot,
q_unsched_react_tot, q_unsched_cap_tot );
pf_cflow_exit(); /* Drop IPC connection */

```

C programs usually have header files “included” somewhere near the beginning after a comment header. Header files may be standard or programmer-created. One standard header file required is `stdio.h` if the program does any I/O operations. Various macros and definitions make up these files. `string.h` is also a standard header file. `cflowlib.h` is a special header file for the CFLOW library. All CFLOW programs must include this file.

All C programs must include a call to `main`, which is where program execution starts.

Immediately after `main` follow a series of declarations of variables local to `main`. The variable `r` is a special CFLOW structure of type `pf_rec`. This is the basic powerflow record structure used to retrieve both input and output data for all types of records.

A three character array representing a two character bus zone code follows. Note that character arrays storing strings should always be declared one byte larger than needed to account for the terminating NULL (`\0`.) Two integer variables, `error` and `status`, are declared next. These will be used to store the return value from calls to `pf_rec_bus`. The variables to collect the zone total quantities are declared as float (single precision real numbers).

`pf_cflow_init` is called to establish the IPC socket connection, which the program will use to communicate with either

the `ipfsrv` or `ipfbat` programs.

We print a question to the terminal window, and use `readln` to retrieve the user selection of a zone to report on, and make sure the string is null-terminated by storing a null in the last element of the array (remember that C indexes array elements from 0 to  $n-1$ ). The input zone id is echoed back in the heading of the report, and the floating point variables are initialized.

Now we begin the actual processing. The first bus record is retrieved by calling `pf_rec_bus` with an action code of “F” (for First). This stores the input record data for the first bus in the currently loaded system in the local structure `r`. But we want the output (solution) quantities. So we call `pf_rec_bus` again, with action code “O” (for Output). The necessary id fields have been stored in `r` by the first call, and these are passed back to `ipfsrv` so it knows what bus you want output values for.

A while loop now executes. The purpose of the while loop is to sequentially access bus records and gather floating point data related to the shunt reactance. `pf_rec_bus` initially used action code “F” to go to the first bus record in the base case. After this, `pf_rec_bus` uses action code “N” to retrieve the next bus record. When the end of all records has been reached, `pf_rec_bus` returns a -1, which causes the while loop to terminate.

Since only records from the user-specified zone are desired in this program, a test is first performed on each bus record to see if it has the correct zone. The program could be made more efficient by not bothering to retrieve output values except for the right buses; however it has been left this way for simplicity in providing an example. If the bus is in the right zone, then the appropriate floating point values are totaled in assignment statements. This program assumes that the zone has no DC buses; if there are any, then the DC solution variables stored in the same fields as the AC shunt would give you weird results, to say the least! Of course, it would be possible to also test for bus type, along with testing for the zone, in order to avoid this problem.

When the loop terminates, the reactance totals are printed to the screen with `printf` statements formatted for decimal output. Then we exit, and release the socket connection, by calling `pf_cflow_exit`.

### 2.11.8 Standard Line Flow Summary Example

The `slfs.c` program was translated from a COPE procedure used by WSCC Technical Staff. It reads an input data file of headings and branches to be reported, looks these up in the currently loaded solved system, and sends the report to a file. The report includes the input headings, and group totals where called for, in an attractive format for printing.

This is what the input data file looks like. Heading lines are those with neither ‘LIN’ or ‘TOT’ on them. They are printed as encountered. ‘LIN’ cards identify a branch to be reported; ‘TOT’ cards call for a total to be printed. The “2” in column 5 indicates that the reverse flow is to be reported.:

```
CANADA AND NORTHWEST
-----
  1. Alberta - British Columbia
LIN LANGDON 500 CBK500 500
LIN LNGDN500 500 CRANBROK 500
LIN LNGDN500 500 CBK500 500
LIN 2 POCA TAP 138 EMC138 138
TOTAL
  2. Canada - Northwest
LIN ING500 500 CUSTER W 500
LIN INGLEDOW 500 CUSTER W 500
LIN 2 NLY230 230 BOUNDARY 230
LIN 2 NELWAY 230 BOUNDARY 230
LIN 2 NLYPHS 230 BOUNDARY 230
LIN SELPHS-1 230 MARSHALL 230
LIN SELPHS-2 230 MARSHALL 230
```

(continues on next page)

(continued from previous page)

```

LIN SELPHS-1 230 BEACON N 230
LIN SELPHS-2 230 BEACON S 230
TOTAL
  3. Northwest - California
LIN MALIN 500 ROUND MT500.
LIN DELTA 115 CASCADE 115.
LIN CAPTJACK 500 OLINDA 500.
TOTAL
  4. Celilo - Sylmar
LIN 2 SYLMAR2I106. SYLMARLA230.
LIN 2 SYLMAR1I106. SYLMARLA230.
LIN 2 SYLMAR2R106. SYLMARLA230.
LIN 2 SYLMAR1R106. SYLMARLA230.
TOTAL
NORTHEAST
-----
  1. MPC High Line
LIN 2 CONRAD 115 CUT BANK115.
LIN GT FALLS 161 HAVRE 161.
.
.
.
```

The program slfs.c prompts the user for the output file name to put the line flow listing in. The input file name is hard-coded in the program.

```

/* slfs.c
  This CFLOW procedure looks up flows and creates a report
  of flows between buses as listed in an input data file.
  Before the CFLOW procedure is called, a solved power flow case
  must be resident in the powerflow server.
  This CFLOW procedure prompts the user for two file names:
  the output file name to put the line flow listing in,
  the input file name of a file to get line data from.
  The input file has a LIN card for each branch to be monitored.
  If there are multiple lines between the same buses, slfs.c
  picks up all lines. A TOT card flags printing of total flow
  since the last TOT card (or since the beginning). Input lines
  without either LIN or TOT are printed directly to the output file.
  Each data card has LIN in columns 1-3 and the Branch identifiers
  in columns 7 to 31. First Bus name and KV in columns 7-18,
  Second bus name and KV in columns 20-31. Column 5 is a flag to
  tell slfs.c whether to use the "Pin" or "Pout" data quantity for
  the total flow. If the flag is "2", then "Pout" is used; otherwise
  "Pin" is used. The flag corresponds to the metering point.
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "cflowlib.h"
/* cf_debug = 1; /* Put this where you want the buffers displayed,for debugging */
int readln( char *s, int lim ) /* Read a line from the terminal input */
```

(continues on next page)

(continued from previous page)

```

{
    int i;
    char c;
    for ( i=0; i < lim - 1 && ( c = getchar() ) != EOF && c != '\n'; ++i )
        s[i] = c;
    s[i] = '\0';
    return i;
}

int freadln( FILE *fp, char *s, int lim ) /* Read a line from a file */
{
    int i;
    char c;
    for ( i=0; i < lim - 1 && ( c = getc( fp ) ) != EOF && c != '\n'; ++i )
        s[i] = c;
    s[i] = '\0';
    return i;
}

main ( int argc, char *argv[] )
{
    pf_comments comments; /* CFLOW structure */
    pf_rec r; /* CFLOW structure */
    FILE *out, *dat;
    char card[82], cout[31];
    char skv1[5], skv2[5];
    char direc;
    int error, lin, tot;
    float brflow, totalflow = 0.0;
    float kv1, kv2;
    pf_cflow_init( argc, argv );
    /* Open the input file. This could be prompted for, but Don says
       he always uses the same input file, which will NOT be in the
       execution directory, so full pathname is supplied.
       If the file cannot be opened, the program terminates
    */
    dat=fopen( "/home/ipf/cflow_progs/slfs.dat", "r" );
    if ( dat == NULL ) {
        printf("Can't open data file\n");
        exit(0);
    }
    /* Prompt the user for the output file name, and open the file.
       Normally, file will be created in the execution directory,
       so cout is only 30 characters.
       If the file cannot be opened, the program terminates.
    */
    printf("Enter output file name > ");
    readln( cout, sizeof(cout) );
    printf("\n");
    out=fopen( cout, "w" );
    if ( out == NULL ) {
        printf("Can't open output file\n");
        exit(0);
    }
}

```

(continues on next page)

(continued from previous page)

```

/* Retrieve caseid and description, and print heading */
pf_rec_comments( &comments, "G" );
fprintf(out, "\n%s\n\n", &comments.h[0][33] );
totalflow = 0.0;
/* Loop for every line in the data file
Read a line from the data file, if not "LIN" or "TOT" then
print the text as is.
If "LIN" retrieve and print line data.
If "TOT" print totalflow and reinitialize it.
*/
/* "C" array indices start at zero; below we use explicit starting index
of 1 for "card" character array (string), so that column numbers
will match array indices.
*/
while ( freadln( dat, &card[1], sizeof( card ) - 1 ) )
{
    int head;

    lin = (!( strcmp( &card[1], "LIN", 3 )) );
    tot = (!( strcmp( &card[1], "TOT", 3 )) );
    head = lin + tot; /* = 0 if not either */
    if (head == 0) /* Heading card - print the text */
    {
        fprintf(out, "%s\n", &card[1]); /* Print from col. 1 - 1st char. is NULL! */
    } /* end if Heading */
    if (lin) /* LIN data card - process line */
    {
        /* Using CFLOW pf_init_branch function; bus kv's must be real numbers */
        strncpy(skv1, &card[15], sizeof(skv1));
        skv1[4] = '\0';
        strncpy(skv2, &card[28], sizeof(skv2));
        skv2[4] = '\0';
        kv1 = atof(skv1);
        kv2 = atof(skv2);

        /* pf_init_branch stores the passed ID fields in r, and zeroes all the other
↪fields */
        pf_init_branch ( &r, "L", &card[7], kv1, &card[20], kv2, "", 0);

        /* Call CFLOW pf_rec_branch to retrieve output solution data */
        error = pf_rec_branch( &r, "O" );
        /* If branch not found, do not print anything. This happens quite often,
↪since
        the input file is canned, and used on all cases. */
        if ( error ) {
            continue; /* This sends it back to the 'while' */
        }
        /* Otherwise, retrieve flows. r is defined in this pgm as a structure of
↪type pf_rec
        i indicates input data, s indicates output data. See cflowlib.h for
↪definitions.
        */
    }
}

```

(continues on next page)



(continued from previous page)

```

    direc = card[5]; /* Meter flag for Pin or Pout */
    if ( direc == '2' )
    { /* meter at second bus */
        brflow = r.s.branch.Pout;
        totalflow += brflow;
        fprintf( out,"%s - %s(M) %s kV num ckts %d %7.1f\n",
                r.i.branch.bus1_name, r.i.branch.bus2_name, skv2,
                r.s.branch.num_ckt, brflow );
    }
    else /* meter at first bus */
    {
        brflow = r.s.branch.Pin;
        totalflow += brflow;
        fprintf( out,"%s(M) - %s %s kV num ckts %d %7.1f \n",
                r.i.branch.bus1_name, r.i.branch.bus2_name, skv2,
                r.s.branch.num_ckt, brflow );
    }
} /* end if LIN */
if (tot) /* TOT data card */
{
    fprintf(out,
            "\n Total flow is %7.1f \n\n",totalflow);
    totalflow = 0.0 ;
} /* end else if TOT */
} /* end while */
pf_cflow_exit();
} /* end main */

```

Note that both input values (r.i.branch.) and output values (r.s.branch.) are reported. The “i” stands for “input” and the “s” stands for “solution”.

The variable cf\_debug is provided for convenience in debugging a CFLOW program. When it is “true” (set non-zero), all the input and output buffers will be dumped to the terminal window, so you can see exactly what your program is sending and getting back. Since this can be very voluminous, you would only want to turn it on in the area where you are having a problem.

### 2.11.9 increm Program

The `increm` program was translated from a COPE procedure used by WSCC Technical Staff. It reads an input data file of branches to be reported, and another file of buses to change generation on. The starting case is hard-coded, and so is the bus that you want to study power transfer from, in this case GADSBY 3 13.8. For each bus in the second file, the generation at GADSBY is increased, that of the other bus decreased, area intertie schedules are adjusted as necessary, the case is solved, and the flow is retrieved for all the branches in the first input file and stored in an array.

The report goes to a file; it consists mostly of a matrix showing the effect of the generation changes on the monitored line flows.

The COPE procedure used the IPS feature INCREM to accomplish this task. IPF has no built-in incrementals function, so the CFLOW program just does what is described in the paragraph above. It is not a general-use incremental program mimicking the IPS function. For simplicity, a lot of names are hard-coded in this program. The user would have to decide whether it would be more efficient to change them in the code for each study, or fix the code to be general and then have to type them in over again for each run.

```

/*****\
UTILITY:    INCREM
TYPE:       Powerflow (IPFSRV v. 209 or later) incremental line flows.
SUMMARY:    Creates an incremental line flow listing of selected lines, sorted
            by areas.
RELATED:
SEE ALSO:   INCREM COPE procedure for IPS
UPDATED:    February 17, 1995
LANGUAGE:   Standard C.  CFLOW libraries.
DEVELOPER:  Walter L. Powell, TEOS, BPA x3234
REQUESTER:
USERS:
PURPOSE:    Creates an incremental line flow listing of selected lines, sorted
            by areas. The incremental flows are computed as the change in flows
            from the base case to the incremental case. Up to 10 incremental
            cases can be submitted, though this limit is easily extended.
\*****/
/* increm.c

```

This CFLOW procedure creates an incremental line flow listing of selected lines, sorted by areas. The incremental flows are computed as the change in flows from the base case to the incremental case.

Several incremental cases can be submitted. The limit here is 10, compared with 64 for the INCREM COPE procedure for IPS. However, this limit can be extended with attendant changes in the incremental storage arrays and in the output reports.

Two aspects makes this CFLOW procedure more complicated than the COPE equivalent.

1. IPF uses area intertie "I" records to define the net area export. If these records are present, an interarea transfer is effected only by changing the scheduled interarea export. If "I" records exist but the particular Areal-Area2 "I" record does not exist, then a new "I" record must be added for Areal-Area2 with an export value of the desired transfer. If no "I" records exist, then the interarea transfer is effected by the ordinary means, namely, by increasing the areal export and decreasing the area2 export.
2. Process INCREM does not exist in IPF. Consequently, the sorting and listing of branches was implemented entirely within this CFLOW procedure.

The code is intentionally batch. It could be made interactive by prompting for file names and bus names. All file names and bus names are hard coded. Changing these requires re-editing, recompiling, and relinking the program. Fortunately, these steps can be performed in a short time.

The program's execution sceniero is as follows.

1. Load in base case history file.

(continues on next page)

(continued from previous page)

```

2. Open branch data file, bus data file, and output report file.
3. Process the branch data file. For each branch, obtain the base case
   line flow.
4. Process the bus data file to identify each transfer pair of buses:
   "busname1" and "busname2". The "busname1" is a hard-coded global
   variable.
   a. For each bus pair, perturb busname1's generation +100MW (and its
      associated areaname1's export +100MW) and busname2's generation
      -100MW (and its associated areaname2 export -100MW).
   b. Solve the case.
   c. Loop through the monitored branches, obtaining the line flows
      for the perturbed case.
5. Print the output report.
   a. Use a branch index array "keysrt" in conjunction with a
      user-written compare routine (to be used with qsort) to obtain a
      double-entry list of monitored branches sorted by the following
      fields: area1, bus1, area2, bus2, id, and section.
   b. Print out the monitored lines flows using the sort index.
*/

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>

#include "cflowlib.h"
#include "ft.h"

#define NULLC    '\0'
#define MAXFLOWS 1000
#define MAXCASES 10
#define TRUE     1
#define FALSE    0
#define LINES_PER_PAGE 64

/* Declare global variables */

struct FLOWS {
    char    bus1[13];          /* bus1, base1 field          */
    char    bus2[13];          /* bus2, base2 field          */
    char    id;                /* id field                    */
    int     section;           /* section field               */
    char    area1[11];          /* area name corresponding to bus1 */
    char    area2[11];          /* area name corresponding to bus2 */
    float    Pin[MAXCASES+1]; /* Tie line flows in MWs      */
    float    Pout[MAXCASES+1]; /* Tie line flows in MWs      */
} br_flow[MAXFLOWS];

/* Declare hard-coded files and bus names */

char *progname = "incrm";
char *basefile = "98hs3inc.bse";

```

(continues on next page)

(continued from previous page)

```

char *branchfile = "line.dat";
char *busname1 = "GADSBY 313.8";
char *busfile = "name2.dat";
char *outfile = "98hs3inc.out";

/* declare function prototypes */

FILE *efopen (char *, char *);
int compare( const void *, const void * );
/* static int compare( const void *key1, const void *key2 ); */
int increment (char *, char *, float, int);
int get_bus_area (char *, char *);
int mod_bus_gen (char *, float);
int mod_area_export (char *, float);
int mod_itie_export (char *, char *, float);
void title (int *, int *, FILE *, pf_comments *, int, char *);

FILE *efopen (char *file, char *mode) /* fopen file, return message */
/* if can't                          */
{
    FILE *fp;

    if ((fp = fopen(file, mode)) != NULL)
        return fp;
    fprintf (stderr, " %s: can't open file %s mode %s\n",
        progname, file, mode);
    return NULL;
}

main(int argc, char *argv[])
{
    pf_comments    c;
    pf_rec         br, itie;

    int i, j, k, status, numcases = 0, keysrt[2*MAXFLOWS], numbr, lineno = 0,
        pageno = 0, intertie_flag = FALSE, section;
    FILE *fp_busfile, *fp_branchfile, *fp_outfile;
    char id, base[5], busname2[13], oldarea[11], string[133], areaname1[11];

    pf_cflow_init( argc, argv);

    /* Load the history data file */

    status = pf_load_oldbase (basefile);
    if (status) {
        fprintf (stderr, " Unable to open history file %s status %d",
            basefile, status);
        pf_cflow_exit();
        return 1;
    }

    /* Determine if any Area Intertie "I" records exist in the base case */

```

(continues on next page)

(continued from previous page)

```

pf_init_rec (&itie, PF_REC); /* Initialize structure prior
                               to calling pf_rec_itie.
                               PF_REC is in header file "ft.h" */
status = pf_rec_itie (&itie, "F");
if (status == 0) intertie_flag = TRUE; /* If success, "I" records exist */

/* Obtain case comments */

pf_init_rec (&c, PF_REC); /* Initialize structure prior
                           to calling pf_rec_comments.
                           PF_REC is in header file "ft.h" */
status = pf_rec_comments (&c, "G");

/* Open the monitored branch data file for read only */

fp_branchfile = fopen(branchfile, "r");
if (fp_branchfile == NULL) {
    fprintf (stderr, " Unable to open monitored branch file %s \n",
            branchfile);
    pf_cflow_exit();
    return 1;
}

/* Open the bus data file for read only */

fp_busfile = fopen(busfile, "r");
if (fp_busfile == NULL) {
    fprintf (stderr, " Unable to open buses list file %s \n",
            busfile);
    pf_cflow_exit();
    return 1;
}

/* Open the output report file for write only */

fp_outfile = fopen(outfile, "w");
if (fp_outfile == NULL) {
    fprintf (stderr, " Unable to open INCREM output report file %s \n",
            outfile);
    pf_cflow_exit();
    return 1;
}

fprintf (fp_outfile, " Base case from history file %s \n", basefile);
for (i=0; i < 3; i++) {
    if (strlen (c.h[i], sizeof(c.h[0])) > 0)
        fprintf (fp_outfile, " %s \n", &c.h[i][1]);
}
/*
Process each record in the monitored branch file. The procedure invokes
the following steps.

```

(continues on next page)

(continued from previous page)

1. Parse the ID fields of each branch entity into the branch data structure.
2. A call to routine "pf\_rec\_branch" with appropriate arguments will obtain the branch output quantities Pin and Pout.
3. Parse the ID fields again of each branch entity into arrays used for sorting and printing the output report. The "area1" and "area2" arrays are obtained indirectly through the zone. First the zone is obtained from the bus data structure of each terminal bus. From each zone, the corresponding area name is obtained via the call "pf\_area\_of\_zone".
4. An slightly unusual convention is employed here for subscript "numbr". Normally, a C-arrays begins with subscript "0". Here, it begins with subscript "1". The reason is that it is necessary to distinguish normal branches from transposed branches in the "keysrt" array, which will be built after all lines have been read in and all incremental cases have been processed. The distinction is done by positive and negative subscripts: +n denotes the normal orientation, -n its transpose. (C cannot distinguish between +0 and -0.)

\*/

```
fprintf (stderr, " Reading in monitored line data \n");
```

```
numbr = 0;
```

```
while ( fgets (string, 132, fp_branchfile) != NULL) {
```

```
    /*
```

```
        Parse the branch data id fields from "string" into structure
        "br.i.branch" and call "pf_rec_branch" with the "0" option to
        retrieve the base case flow.
```

```
        Copy the parsed branch data id fields into a second structure
        "br_flow[]" for storing the flow results.
```

```
        Note that the strncpy command requires that the strings be
        explicitly null-terminated.
```

```
    */
```

```
    pf_init_rec (&br, PF_REC);    /* Initialize structure prior
                                   to calling pf_rec_branch.
                                   PF_REC is in header file "ft.h" */
```

```
    strcpy (br.i.branch.type, "L ");
    strncpy (br.i.branch.bus1_name, &string[6], 8);
    br.i.branch.bus1_name[8] = NULLC;
    strncpy (base, &string[14], 4);
    base[4] = NULLC;
    br.i.branch.bus1_kv = atof (base);
    strncpy (br.i.branch.bus2_name, &string[19], 8);
    br.i.branch.bus2_name[8] = NULLC;
    strncpy (base, &string[27], 4);
```

(continues on next page)

(continued from previous page)

```

base[4] = NULLC;
br.i.branch.bus2_kv = atof (base);
id = string[31];
if (id == '\n' || id == '\t' || id == '\0') id = ' ';
br.i.branch.ckt_id = id;
if (strlen (string) > 32) {
    base[0] = string[32];
    base[1] = NULLC;
    section = atoi (base);
} else {
    section = 0;
}
br.i.branch.section = section;
/*
The branch data is now loaded in structure "branch". A call to
"pf_rec_branch" will retrieve the desired information.
*/
status = pf_rec_branch (&br, "0");
if (!status) {
    numbr++;
    strcpy (br_flow[numbr].bus1, &string[6]);
    br_flow[numbr].bus1[12] = NULLC;
    strcpy (br_flow[numbr].bus2, &string[19]);
    br_flow[numbr].bus2[12] = NULLC;
    br_flow[numbr].id = br.i.branch.ckt_id;
    br_flow[numbr].section = br.i.branch.section;
    br_flow[numbr].Pin[0] = br.s.branch.Pin;
    br_flow[numbr].Pout[0] = br.s.branch.Pout;

    /* Get the area name associated with bus1 indirectly through the
       zone of bus1 */

    status = get_bus_area (br_flow[numbr].bus1, br_flow[numbr].area1);
    if (status) {
        fprintf (stderr, " Unable to retrieve area associated with bus1 of \
monitored branch %s %s \n", br_flow[numbr].bus1, br_flow[numbr].bus2);
        pf_cflow_exit();
        return 1;
    }

    /* Get the area name associated with bus2 indirectly through the
       zone of bus2 */

    status = get_bus_area (br_flow[numbr].bus2, br_flow[numbr].area2);
    if (status) {
        fprintf (stderr, " Unable to retrieve area associated with bus2 of \
monitored branch %s %s \n", br_flow[numbr].bus1, br_flow[numbr].bus2);
        pf_cflow_exit();
        return 1;
    }
}
}

```

(continues on next page)

(continued from previous page)

```

/* Begin the incremental bus loop. This is the list of buses in
   testdc3.bdat, processed one-by-one. */

while ( fgets (string, 80, fp_busfile) != NULL) {

    strncpy (busname2, string, sizeof(busname2));
    busname2[12] = NULLC;
    status = increment (busname1, busname2, 100.0, intertie_flag);
    if (!status) {

        numcases++;

        /* Solve the modified case */

        status = pf_solution ();

        /* Write change case comments */

        if (status) {

            fprintf (fp_outfile, "\n Incremental case number %d failed \n",
                     numcases);
            fprintf (fp_outfile, "      100 MW scheduled from \"%s\" to \"%s\" \n",
                     busname1, busname2);
            fprintf (stderr, " Incremental case number %d failed \n",
                     numcases);
            for (i = 1; i <= numbr; i++) {
                br_flow[i].Pin[numcases] = 0.0;
                br_flow[i].Pout[numcases] = 0.0;
            }
        } else {

            fprintf (fp_outfile, "\n Incremental case number %d \n", numcases);
            fprintf (fp_outfile, "      100 MW scheduled from \"%s\" to \"%s\" \n",
                     busname1, busname2);
            fprintf (stderr, "\n Incremental case number %d \n", numcases);
            fprintf (stderr, "      100 MW scheduled from \"%s\" to \"%s\" \n",
                     busname1, busname2);

            for (i = 1; i <= numbr; i++) {
                pf_init_rec (&br, PF_REC); /* Initialize structure prior
                                             to calling pf_rec_branch.
                                             PF_REC is in header file "ft.h" */

                strcpy (br.i.branch.type, "L ");
                strncpy (br.i.branch.bus1_name, br_flow[i].bus1, 8);
                br.i.branch.bus1_name[8] = NULLC;
                br.i.branch.bus1_kv = atof (&br_flow[i].bus1[8]);
                strncpy (br.i.branch.bus2_name, br_flow[i].bus2, 8);
                br.i.branch.bus2_name[8] = NULLC;
                br.i.branch.bus2_kv = atof (&br_flow[i].bus2[8]);
                br.i.branch.ckt_id = br_flow[i].id;
            }
        }
    }
}

```

(continues on next page)



(continued from previous page)

```

        status = pf_rec_branch (&br, "0");
        br_flow[i].Pin[numcases] = br.s.branch.Pin;
        br_flow[i].Pout[numcases] = br.s.branch.Pout;
    }
}
}
}

/* Obtain sorted double entry index "keysrt" for branch flows
   keysrt[i] > 0 means "br_flow" is processed as is
   < 0 means "br_flow" is processed transposed
*/

for (i = 0; i < numbr; i++) {
    keysrt[2*i] = i + 1;
    keysrt[2*i+1] = -keysrt[2*i];
}
qsort ( keysrt, 2*numbr, sizeof (keysrt[0]), compare );

title (&lineno, &pageno, fp_outfile, &c, numcases, "");

oldarea[0] = NULLC;
for (i = 0; i < 2*numbr; i++) {
    j = keysrt[i];
    if (j > 0) {
        strcpy (areaname1, br_flow[j].area1);
    } else {
        strcpy (areaname1, br_flow[-j].area2);
    }
    if (strcmp (areaname1, oldarea) != 0) {
        if (lineno+4 > LINES_PER_PAGE) {
            lineno = 0;
            title (&lineno, &pageno, fp_outfile, &c, numcases, areaname1);
        } else {
            fprintf (fp_outfile, "\n From area %s \n\n", areaname1);
            lineno += 3;
        }
        strcpy (oldarea, areaname1);
    }
    if (lineno+1 > LINES_PER_PAGE) {
        lineno = 0;
        title (&lineno, &pageno, fp_outfile, &c, numcases, areaname1);
    }
    if (j > 0) {
        fprintf (fp_outfile, " %s %s %c %s %8.2f  ", br_flow[j].bus1,
            br_flow[j].bus2, br_flow[j].id, br_flow[j].area2,
            br_flow[j].Pin[0]);
        for (k = 1; k <= numcases; k++) {
            fprintf (fp_outfile, " %7.2f", br_flow[j].Pin[k] - br_flow[j].Pin[0]);
        }
    } else {
        fprintf (fp_outfile, " %s %s %c %s %8.2f  ", br_flow[-j].bus2,

```

(continues on next page)

(continued from previous page)

```

        br_flow[-j].bus1, br_flow[-j].id, br_flow[-j].areal,
        -br_flow[-j].Pout[0]);
    for (k = 1; k <= numcases; k++) {
        fprintf (fp_outfile, " %7.2f",
            -br_flow[-j].Pout[k] + br_flow[-j].Pout[0]);
    }
}
fprintf (fp_outfile, "\n");
lineno++;
}

pf_cflow_exit();
fclose (fp_busfile);
fclose (fp_branchfile);
fclose (fp_outfile);
if (!status) {
    exit (0);
} else {
    fprintf (stderr, " %s aborted with errors \n", progname);
    exit (1);
}
}

int get_bus_area (char *name, char *area)

/* This routine returns the area name given the bus name. It is obtained
indirectly through the bus bus data structure. First, essential information
is inserted into the bus data structure and the remaining data in the
structure is obtained after calling "pf_rec_bus". From the zone name in
the bus structure, the area name is obtained by calling "pf_area_of_zone".
*/

{
    pf_rec bus;
    char base[5], zone[3];
    int len, status;

    pf_init_rec (&bus, PF_REC); /* Initialize structure prior
                                to calling pf_rec_bus.
                                PF_REC is in header file "ft.h" */

    strcpy (bus.i.ACbus.type, "B ");
    strncpy (bus.i.ACbus.name, name, 8);
    bus.i.ACbus.name[8] = NULLC;
    strcpy (base, &name[8]);
    bus.i.ACbus.kv = atof (base);
    status = pf_rec_bus (&bus, "G");
    if (status) {
        fprintf (stderr, " Bus %s is not in history file %s \n", name, basefile);
        return 1;
    }
    strncpy (zone, bus.i.ACbus.zone, 2);
    zone[2] = NULLC;

```

(continues on next page)

(continued from previous page)

```

status = pf_area_of_zone (area, zone);
if (status) {
    fprintf (stderr, " No area data in history file %s \n", basefile);
    return 1;
}
return 0;
}

int increment (char *busname1, char *busname2, float delta_gen, int flag)

/* This routine applies +/- delta_gen to pairs of buses, areas, and/or
   intertie records to effect the desired transfer. */

{
    pf_rec      b, a, itie;
    char areaname1[11], areaname2[11];
    int status;

    /* The following are declared "static" such that they can be initialized
       to NULL, are local, and are not volatile, i.e., they retain their value
       after the routine is executed. In FORTRANese, they are similar to local
       variables, initialized with a DATA statement, and retained with a
       SAVE statement.
    */
    static char oldbus1[13] = "", oldarea1[11] = "", oldbus2[13] = "",
               oldarea2[13] = "";

    /* Get areaname1 associated with busname1 */

    status = get_bus_area (busname1, areaname1);
    if (status) {
        fprintf (stderr, " Unable to retrieve area associated with bus %s \n",
            busname1);
        pf_cflow_exit();
        return 1;
    }

    /* Get areaname2 associated with busname2 */

    status = get_bus_area (busname2, areaname2);
    if (status) {
        fprintf (stderr, " Unable to retrieve area associated with bus %s \n",
            busname2);
        pf_cflow_exit();
        return 1;
    }

    /* Restore original Pgen on "oldbus1" and modify Pgen on "busname1".
       In this case, restoring the original Pgen on "oldbus1" is not
       necessary because it never changes its name once it is assigned.
       The same is true for area "oldarea1" and for intertie "oldarea1-oldarea2".
       This extra logic (two lines in each instance) is added for generality.
    */

```

(continues on next page)

(continued from previous page)

```

*/

if (strcmp (busname1, oldbus1) != 0) {
    if (strlen (oldbus1) > 0)
        status = mod_bus_gen (oldbus1, -delta_gen);
    status = mod_bus_gen (busname1, delta_gen);
}

/* Restore original Pgen on oldbus2 and modify Pgen on busname2 */

if (strcmp (busname2, oldbus2) != 0) {
    if (strlen (oldbus2) > 0)
        status = mod_bus_gen (oldbus2, delta_gen);
    status = mod_bus_gen (busname2, -delta_gen);
}
if (flag) {

    /* Area Intertie "I" records exist and these records define the net
       area interchange export. Restore the original export on intertie
       "oldarea1-oldarea2" and modify the export on intertie
       "areaname1-areaname2" */

    if (strlen (oldarea1) > 0 && strlen (oldarea2) > 0)
        status = mod_itie_export (oldarea1, oldarea2, -delta_gen);
    status = mod_itie_export (areaname1, areaname2, delta_gen);
} else {

    /* Area intertie "I" records do not exist. Work directly on the
       area records. Modify export on each area "areaname1" and
       "areaname2" */

    if (strlen (oldarea1) > 0)
        status = mod_area_export (oldarea1, -delta_gen);
    status = mod_area_export (areaname1, delta_gen);
    if (strlen (oldarea2) > 0)
        status = mod_area_export (oldarea2, delta_gen);
    status = mod_area_export (areaname2, -delta_gen);
}
strcpy (oldbus1, busname1);
strcpy (oldbus2, busname2);
strcpy (oldarea1, areaname1);
strcpy (oldarea2, areaname2);
return status;
}

int mod_bus_gen (char *busname, float delta_gen)

/* This routine changes Pgen on bus "busname" an amount "delta_pgen" */

{
    pf_rec      b;
    int status;

```

(continues on next page)

(continued from previous page)

```

pf_init_rec (&b, PF_REC); /* Initialize structure prior
                           to calling pf_rec_bus.
                           PF_REC is in header file "ft.h" */
strcpy (b.i.ACbus.type, "B ");
strncpy (b.i.ACbus.name, busname, 8);
b.i.ACbus.name[8] = NULLC;
b.i.ACbus.kv = atof (&busname[8]);

status = pf_rec_bus (&b, "G");
if (status) {
    fprintf (stderr, " Bus %s is not in history file %s \n",
             busname, basefile);
    pf_cflow_exit();
    return 1;
}

/* Add "delta_gen" to bus generation */

b.i.ACbus.Pgen = b.i.ACbus.Pgen + delta_gen;
status = pf_rec_bus (&b, "M");
return status;
}

int mod_area_export (char *areaname1, float delta_export)

/* This routine changes Export on Area "areaname1" an amount "delta_export" */

{
    pf_rec      a;
    int status;

    pf_init_rec (&a, PF_REC); /* Initialize structure prior
                               to calling pf_rec_bus.
                               PF_REC is in header file "ft.h" */
    strcpy (a.i.area.type, "A ");
    strcpy (a.i.area.name, areaname1);
    status = pf_rec_area (&a, "G");
    if (status) {
        fprintf (stderr, " Area %s is not in history file %s \n",
                 a.i.area.name, basefile);
        pf_cflow_exit();
        return 1;
    }

    /* Add "delta_export" to area export */

    a.i.area.sched_export = a.i.area.sched_export + delta_export;
    status = pf_rec_area (&a, "M");
    return status;
}

```

(continues on next page)

(continued from previous page)

```

int mod_itie_export (char *areaname1, char *areaname2, float delta_export)

/* This routine changes Export on area intertie "areaname1-areaname2" and
   amount "delta_export" */

{
    pf_rec      itie;
    int status;

    pf_init_rec (&itie, PF_REC); /* Initialize structure prior
                                   to calling pf_rec_itie.
                                   PF_REC is in header file "ft.h" */
    if (strcmp (areaname1, areaname2) == 0) {
        return 0;
    } else {
        strcpy (itie.i.itie.type, "I ");
        strcpy (itie.i.itie.area1_name, areaname1);
        strcpy (itie.i.itie.area2_name, areaname2);
        status = pf_rec_itie (&itie, "G");
        if (status) {
            itie.i.itie.sched_export = delta_export;
            status = pf_rec_itie (&itie, "A"); /* Add a new "I" record */
        } else {
            itie.i.itie.sched_export = itie.i.itie.sched_export + delta_export;
            status = pf_rec_itie (&itie, "M"); /* Modify existing "I" record */
        }
        return status;
    }
}

int compare( const void *key1, const void *key2 )
/* static int compare( const void *key1, const void *key2 ) */

/* This comparing function obtains the relative sort order of *key1 and
   *key2 upon the following fields:

    1. area1.
    2. bus1.
    3. area2.
    4. bus2.
    5. id.

*/

{
    int i = *((int *) key1), j = *((int *) key2), comp;
    char *area11, *area12, *area21, *area22, *bus11, *bus12, *bus21, *bus22,
        id1, id2;
    int sect1, sect2;

    if (i == j) {
        return 0;
    }

```

(continues on next page)

(continued from previous page)

```

} else {
    if (i > 0) {
        area11 = br_flow[i].area1;
        area12 = br_flow[i].area2;
        bus11 = br_flow[i].bus1;
        bus12 = br_flow[i].bus2;
        id1 = br_flow[i].id;
        sect1 = br_flow[i].section;
    } else {
        area11 = br_flow[-i].area2;
        area12 = br_flow[-i].area1;
        bus11 = br_flow[-i].bus2;
        bus12 = br_flow[-i].bus1;
        id1 = br_flow[-i].id;
        sect1 = br_flow[-i].section;
    }
    if (j > 0) {
        area21 = br_flow[j].area1;
        area22 = br_flow[j].area2;
        bus21 = br_flow[j].bus1;
        bus22 = br_flow[j].bus2;
        id2 = br_flow[j].id;
        sect2 = br_flow[j].section;
    } else {
        area21 = br_flow[-j].area2;
        area22 = br_flow[-j].area1;
        bus21 = br_flow[-j].bus2;
        bus22 = br_flow[-j].bus1;
        id2 = br_flow[-j].id;
        sect2 = br_flow[-j].section;
    }
    comp = strcmp (area11, area21);
    if (comp == 0) comp = strcmp (bus11, bus21);
    if (comp == 0) comp = strcmp (area12, area22);
    if (comp == 0) comp = strcmp (bus12, bus22);
    if (comp == 0) comp = (unsigned int)id1 - (unsigned int)id2;
    if (comp == 0) comp = sect1 - sect2;

    return comp;
}
}

void title ( int *lineno, int *pageno, FILE *fp_outfile, pf_comments *c,
            int numcases, char *areaname)
{
    /* Write base case comments */

    int i;

    fprintf (fp_outfile, "\f\n\n");
    (*pageno)++;
    for (i=0; i < 3; i++) {

```

(continues on next page)

(continued from previous page)

```

    if (strlen (c->h[i]) > 0)
        fprintf (fp_outfile, " %s \n", &c->h[i][1]);
        (*lineno)++;
    }
    fprintf (fp_outfile, "\n INCREMENTAL report (DELTA) line flows in MW \
page No. %d \n\n", *pageno);
    fprintf (fp_outfile, " From bus      To bus      cir To area      Base Flow      ");
    for (i = 1; i <= numcases; i++) {
        fprintf (fp_outfile, "Incr %2d ", i);
    }
    (*lineno) += 6;
    if (strlen (areaname) > 0) {
        fprintf (fp_outfile, "\n From area %s \n\n", areaname);
        (*lineno) += 3;
    }
}

```

### 2.11.10 libcf flow API Reference

The header file (`cflowlib.h`) used both by all CFLOW programs to define structures, constants, etc. in the `libcf flow` library.

union **input\_data**

*#include <cflowlib.h>* Input data union.

#### Public Members

*pf\_AC\_bus* **ACbus**

Use for accessing AC bus input data.

*pf\_DC\_bus* **DCbus**

Use for accessing DC bus input data.

*pf\_branch* **branch**

Use for accessing branch input data.

*pf\_LD* **LD**

Use for accessing Two Terminal DC Line input data.

*pf\_LM* **LM**

Use for accessing Multiterminal DC Line input data.

*pf\_E* **E**

Use for accessing Equivalent Transmission Line Branch input data.



*pf\_L* **L**

Use for accessing Balanced Transmission Line input data.

*pf\_T* **T**

Use for accessing Transformer input data.

*pf\_TP* **TP**

Use for accessing Phase Shifting Transformer input data.

*pf\_R* **R**

Use for accessing Regulating Transformer (LTC voltage control) input data.

*pf\_RN* **RN**

Use for accessing Regulating Transformer (Constraints on VAR flow via change to RQ type of limits are violated) input data.

*pf\_RQ* **RQ**

Use for accessing Regulating Transformer (LTC VAR control) input data.

*pf\_RV* **RV**

Use for accessing Regulating Transformer (Constraints on VAR flow via change to RP type of limits are violated) input data.

*pf\_RP* **RP**

Use for accessing Regulating Transformer (LTC phase shifter) input data.

*pf\_RM* **RM**

Use for accessing Regulating Transformer input data.

*pf\_RZ* **RZ**

Use for accessing Series Compensated Rani Model input data.

*pf\_area* **area**

Use for accessing area input data.

*pf\_itie* **itie**

Use for accessing intertie input data.

*pf\_cbus* **cbus**

Use for accessing continuation bus input data.

*pf\_qcurve* **qcurve**

Use for accessing reactive power capability curve input data.

*pf\_xdata* **xdata**

Use for accessing switched reactance input data.

char **cmnt**[120]

Use for accessing comment.

struct **pf\_AC\_bus**

*#include <cfwlib.h>* This structure holds power flow input data for an AC bus. The fields type, owner, name, kV, and zone are the same for both AC and DC buses.

### Public Members

char **type**[3]

Two character bus record type, for example, B, BS, BQ, etc.

char **owner**[4]

Three character bus owner.

char **name**[9]

Eight character bus name.

float **kv**

Base kV of the bus.

char **zone**[3]

Two character zone name.

int **dummy1**

Ignore. This is a dummy field used for alignment purposes.

float **Pload**

Real load in MW.

float **Qload**

Reactive load in MVAR.

float **Pshunt**

Real shunt in MW.

float **Qshunt**

Reactive shunt in MVAR.

float **Pmax**

Maximum real load in MW.

float **Pgen**

Scheduled real power in MW.

float **Qsch\_Qmax**

Scheduled reactive load in MVAR (Qsch) or a real number designating maximum reactive power in MVAR (Qmax).

float **Qmin**

Minimum reactive power in MVAR.

float **Vhold\_Vmax**

Voltage to hold in per unit (Vhold) or a real number designating a maximum voltage limit in per unit (Vmax), depending on the bus type.

float **Vmin\_Vdeg**

Minimum voltage limit in per unit, or voltage angle for the BS bus

char **rmt\_name**[9]

Eight character remote bus name.

float **rmt\_kv**

Base kV of a remote bus.

char **dummy2**

Ignore. This is a dummy field used for alignment purposes.

float **pct\_vars**

Percent vars supplied for control of remote bus.

struct **pf\_area**

*#include <cfowlib.h>* This structure holds power flow input data for control areas which can be made up of multiple zones.

## Public Members

char **type**[3]

Record type; here “A” for area record.

char **name**[11]

Ten character area name.

char **sbus\_name**[9]

Eight character area slack bus name.

float **sbus\_kv**

Base kV of the area slack bus.

float **sched\_export**

Scheduled export power from an area.

char **zone0**[3]

Two character zone defined to be in an area.

char **zone1**[3]

Two character zone defined to be in an area.

char **zone2**[3]

Two character zone defined to be in an area.

char **zone3**[3]

Two character zone defined to be in an area.

char **zone4**[3]

Two character zone defined to be in an area.

char **zone5**[3]

Two character zone defined to be in an area.

char **zone6**[3]

Two character zone defined to be in an area.

char **zone7**[3]

Two character zone defined to be in an area.

char **zone8**[3]

Two character zone defined to be in an area.

char **zone9**[3]

Two character zone defined to be in an area.

float **max\_Vpu**

Maximum per unit voltage.

float **min\_Vpu**

Minimum per unit voltage.

struct **pf\_area\_soln**

*#include <cflowlib.h>* This structure holds power flow solution (output) data for control areas which can be made up of multiple zones.

## Public Members

char **type**[3]

Two character record type.

float **Pgen**

Total area generation.

float **Pload**

Total area load.

float **Ploss**

Total area losses.

float **Pexport**

Total actual export.

struct **pf\_branch**

*#include <cfowlib.h>* This structure holds power flow input data for a branch record.

## Public Members

char **type**[3]

Two character branch record type.

char **owner**[4]

Three character branch owner.

char **bus1\_name**[9]

Bus 1 name.

float **bus1\_kv**

Base kV for bus 1.

int **meter**

The line metering point for area tie lines.

1 = bus 1 end.

2 = bus 2 end.

If blank, metering point will be identified (1) by location where line ownership differs from bus ownership or (2) when buses at end of tie line have same ownership, then the bus name 1 will be the metering point.

char **bus2\_name**[9]

Bus 2 name.

float **bus2\_kv**

Base kV for bus 2.

char **ckt\_id**

Circuit identification if more than one parallel branch exists.

int **section**

Section number for making an equivalent for series elements (numeric). Program assembles series elements in numerical order of section numbers (need not be consecutive).

float **total\_rating**

Total ampere rating for all lines.

int **num\_ckts**

Number of parallel circuits represented by this record, for information purposes only. The equivalent impedance is entered in r, x, g, b for lines with equal legs.

float **r**

Per unit resistance at base kV and base MVA.

float **x**

Per unit reactance at base kV and base MVA.

float **g**

Per unit conductance  $G_{pi}/2$  at base kV and MVA. This format is for balanced lines when  $Y_{pi}$  sending equals  $Y_{pi}$  receiving and only  $Y_{pi}$  needs to be input.

float **b**

Per unit susceptance  $B_{pi}/2$  at base kV and MVA.

float **tap1**

Tap at bus1 of transformer, in kV for normal tap, in degrees for phase shifter.

float **tap2**

Tap at bus2 of transformer, in kV.

float **alpha\_N\_deg**

Initial firing angle in degrees at rectifier (DC lines).

float **gamma\_0\_deg**

Minimum margin angle in degrees at inverter (DC lines).

char **descrip**[9]

Unknown.

char **date\_in**[4]

Energization Date in format “MYYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

char **date\_out**[4]

De-nergization Date in format “MYYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

float **thermal\_rating**

Thermal rating in Amps.

float **bottleneck\_rating**

Bottleneck rating in Amps.

float **emergency\_rating**

Emergency rating in MVA.

struct **pf\_branch\_soln**

*#include <cfwlib.h>* This structure holds power flow solution data for branch records.

## Public Members

char **type**[3]

Two character record type.

int **num\_cks**

If ckt id was “\*”, contains number of parallels

float **Pin**

Real power flow at bus1, in MW. Positive indicates flow from bus1 toward bus2.

float **Qin**

Reactive power flow at bus1, in MVAR. Positive indicates flow from bus1 toward bus2.

float **Pout**

Real power flow at bus2, in MW. Negative indicates flow from bus1 to bus2 (i.e., without losses, Pout = -Pin).

float **Qout**

Reactive power flow at bus2, in MVAR. Negative indicates flow from bus1 to bus2.

float **Ploss**

Real losses, in MW (Pin + Pout).

float **Qloss**

Reactive losses, in Mvar ( $Q_{in} + Q_{out}$ ).

float **crit\_line\_load\_amps**

Largest current in any section of a line (amps).

float **crit\_line\_rat\_amps**

Actual value of rating used (amps).

char **crit\_line\_rat\_code**

Type of rating used – N, B, T, or E.

int **crit\_line\_load\_term**

Largest flow in a transformer terminal (MVA).

float **crit\_xfmr\_load\_mva**

Largest flow in a transformer (MVA).

float **crit\_xfmr\_rat\_mva**

Actual value of rating used (MVA).

char **crit\_xfmr\_rat\_code**

Type of rating used – N, B, T, or E.

int **crit\_xfmr\_load\_term**

Largest flow in a transformer terminal (MVA).

float **tot\_line\_load\_pct**

Percent line compensation (total negative reactance divided by total positive reactance).

float **tot\_line\_load\_amps**

Line compensation in amps.

float **tot\_xfmr\_load\_pct**

Percent loading on the transformer, using the indicated rating

float **tot\_xfmr\_load\_mva**

Transformer load in apparent power (MVA).

float **tap1**

Final tap at bus1 of transformer, in kV for normal tap, in degrees for phase shifter.

float **tap2**

Final tap at bus2 of transformer, in kV.



struct **pf\_bus\_AC\_soln**

*#include <cfowlib.h>* This struct stores power flow solution values for an AC bus.

### Public Members

char **type**[3]

Two character bus record type, for example, B, BS, BQ, etc.

float **Pgen**

Solved real power generation in MW.

float **Qgen**

Solved reactive power in MVAR.

float **Vmag**

Solved voltage magnitude in per unit.

float **Vdeg**

Solved voltage angle in degrees.

float **Pload**

Solved real load in MW.

float **Qload**

Solved reactive load in MVAR.

float **Bshunt\_used**

Total shunt used, net of capacitors (+) and reactors (-).

float **Bshunt\_sch**

Total shunt available, net of capacitors and reactors.

float **Bshunt\_used\_cap**

Capacitive shunt used, MVAR.

float **Bshunt\_sch\_cap**

Capacitive shunt available, MVAR.

float **Bshunt\_used\_rx**

Reactive shunt used, MVAR.

float **Bshunt\_sch\_rx**

Reactive shunt available, MVAR.

float **Qunsch**

MVARs produced, on a type BS or BE bus.

struct **pf\_bus\_DC\_soln**

*#include <cfowlib.h>* This struct stores power flow solution values for a DC bus.

### Public Members

char **type**[3]

Two character bus record type.

float **P\_DC**

AC real power into the DC bus, positive at the rectifier and negative at the inverter.

float **Q\_DC**

AC reactive power into the DC bus, returned as a positive number.

float **V\_DC**

DC terminal voltage (final voltage at commutating bus, in kV).

float **converter\_deg**

Converter angle, alpha for rectifier, gamma for inverter, in degrees

float **P\_valve\_losses**

Difference between AC power and DC power.

float **Q\_valve\_losses**

Difference between AC power and DC power (same as Q\_DC).

float **dummy1**

Ignore. This is a dummy field used for alignment purposes.

float **dummy2**

Ignore. This is a dummy field used for alignment purposes.

float **dummy3**

Ignore. This is a dummy field used for alignment purposes.

float **dummy4**

Ignore. This is a dummy field used for alignment purposes.

float **dummy5**

Ignore. This is a dummy field used for alignment purposes.

float **dummy6**

Ignore. This is a dummy field used for alignment purposes.

float **dummy7**

Ignore. This is a dummy field used for alignment purposes.

struct **pf\_case\_stats**

*#include <cfowlib.h>* Miscellaneous case data.

## Public Members

char **PF\_version**[11]

Ten character string containing Powerflow version information.

float **base\_mva**

Base MVA of the base case (normally 100.0).

int **num\_DC\_systems**

An integer count of the number of DC systems in the case.

int **num\_areas**

An integer count of the number of areas in the case.

int **num\_ities**

An integer count of the number of interties in the case.

int **num\_zones**

An integer count of the number of zones in the case.

int **num\_owners**

An integer count of the number of owners in the case.

int **num\_buses**

An integer count of the number of buses in the case (both AC and DC)

int **num\_area\_slack\_buses**

An integer count of the number of area slack buses in the case.

int **num\_DC\_buses**

An integer count of the number of dc buses in the case.

int **num\_AGC\_buses**

An integer count of the number of buses with AGC control in the case.

int **num\_BX\_buses**

An integer count of the number of BX (constant V using switched Q) buses in the case.

int **num\_adjustable\_buses**

An integer count of the number of adjustable buses in the case.

int **num\_pct\_var\_ctrl\_buses**

An integer count of the number of buses with percent VAR control in the case.

int **num\_branches**

An integer count of the number of branches in the case.

int **num\_circuits**

An integer count of the number of circuits in the case. All parallel lines count as one circuit.

int **num\_DC\_lines**

An integer count of the number of DC lines in the case.

int **num\_LTC\_xfmr**

An integer count of the number of LTC transformers in the case.

int **num\_phase\_shifters**

An integer count of the number of phase shifters in the case.

int **case\_soln\_status**

An integer containing the solution status. Corresponds to enumerated variables as follows:

1 = NO\_CASE (no case data loaded)

2 = UNSOLVED (netdata loaded)

5 = SOLVED (successful solution, or solved case loaded)

6 = SAVED (solved case has been saved)

7 = DIVERGED (unsuccessful solution - diverged)

int **num\_diff\_kv**

An integer count of the number of unique kVs in the case.

int **num\_rec\_types**

An integer count of the number of unique record types in the case.

struct **pf\_cbus**

*#include <cfowlib.h>* This structure holds power flow input data for continuation bus (+) records.

## Public Members

char **type**[3]

Two character bus record type, for example, “+”, “+A”, etc.

char **owner**[4]

Three character bus owner.

char **name**[9]

Eight character bus name.

float **kv**

Base kV of the bus.

char **code\_year**[3]

Two character extension of type.

float **Pload**

Real load in MW belonging to this owner.

float **Qload**

Reactive load in MVAR belonging to this owner.

float **Gshunt**

Fixed real shunt in MW.

float **Bshunt**

Fixed reactive shunt in MVAR.

float **Pgen**

Scheduled real power in MW for this owner.

float **Qgen\_Qmax**

Scheduled reactive power in MVAR (Qgen) or maximum reactive power in MVAR (Qmax).

float **Qmin**

Minimum reactive power in MVAR.

struct **pf\_cbus\_soln**

*#include <cfwlib.h>* This structure holds power flow solution (output) data for continuation bus (+) records.

### Public Members

char **type**[3]

Two character bus record type, for example, “+”, “+A”, etc.

float **Pgen**

Solved real power generation in MW.

float **Qgen**

Solved reactive power generation in MVAR.

float **Pload**

Solved real load (same as input).

float **Qload**

Solved reactive load (same as input).

float **Gshunt**

Solved real shunt.

float **Bshunt**

Solved reactive shunt.

struct **pf\_comments**

*#include <cflowlib.h>* Comment card data.

### Public Members

char **case\_name**[11]

Ten character string containing caseid.

char **case\_desc**[21]

Twenty character string containing case description.

char **h**[3][133]

A character array containing case headers. The first one, **h**[0], is generated by IPF, and contains the program version, the caseid and description, and the date of the run. The other two are user-specified.

char **c**[20][121]

A character array containing case comments.

struct **pf\_DC\_bus**

*#include <cflowlib.h>* This structure holds power flow input data for a DC bus. The fields **type**, **owner**, **name**, **kV**, and **zone** are the same for both AC and DC buses.

## Public Members

char **type**[3]

Two character bus record type.

char **owner**[4]

Three character bus owner.

char **name**[9]

Eight character bus name.

float **kv**

Base kV of the bus.

char **zone**[3]

Two character zone name.

int **bridges\_per\_ckt**

Number of dc bridges per circuit.

float **smooth\_rx\_mh**

Smoothing reactance in millihenries (mH).

float **alpha\_min\_deg**

Alpha\_min in degrees.

float **alpha\_stop\_deg**

Alpha\_stop in degrees.

float **valve\_drop\_per\_bridge\_volts**

Voltage drop per valve.

float **bridge\_current\_rating\_amps**

DC current rating.

float **alpha\_gamma\_N\_deg**

Alpha\_N or gamma\_N in degrees.

float **gamma\_0\_deg**

Gamma\_0 in degrees.

float **P\_sched**

Scheduled power MW.

float **V\_sched**

Scheduled voltage kV.

float **dummy1**

Ignore. This is a dummy field used for alignment purposes.

char **commutating\_bus\_name**[9]

Eight character commutating bus name.

float **commutating\_bus\_kv**

Commutating bus kV.

char **converter\_code**

One character converter code.

float **dummy2**

Ignore. This is a dummy field used for alignment purposes.

struct **pf\_E**

*#include <cflowlib.h>* This structure holds power flow input data for a equivalent transmission line record (asymmetric pi representation).

### Public Members

char **type**[3]

Two character record type “E”.

char **owner**[4]

Three character owner.

char **bus1\_name**[9]

Bus 1 name.

float **bus1\_kv**

Base kV for bus 1.

int **meter**

The line metering point for area tie lines.

1 = bus 1 end.

2 = bus 2 end.

If blank, metering point will be identified (1) by location where line ownership differs from bus ownership or (2) when buses at end of tie line have same ownership, then the bus name 1 will be the metering point.

char **bus2\_name**[9]

Bus 2 name.



float **bus2\_kv**

Base kV for bus 2.

char **ckt\_id**

Circuit identification if more than one parallel line exists.

int **section**

Section number for making an equivalent for series elements (numeric). Program assembles series elements in numerical order of section numbers (need not be consecutive).

float **total\_rating**

Total ampere rating for all lines.

int **num\_ckts**

Number of parallel circuits represented by this record, for information purposes only. The impedance is entered in r, x, g1, b1, g2, b2.

float **r**

Per unit resistance at base kV and base MVA.

float **x**

Per unit reactance at base kV and base MVA.

float **g1**

Per unit conductance at base kV and MVA for bus 1 end of line.

float **b1**

Per unit susceptance at base kV and MVA for bus 1 end of line.

float **g2**

Per unit conductance at base kV and MVA for bus 2 end of line.

float **b2**

Per unit susceptance at base kV and MVA for bus 2 end of line..

float **dummy1**

Ignore. This is a dummy field used for alignment purposes.

float **dummy2**

Ignore. This is a dummy field used for alignment purposes.

char **dummy3**[9]

Ignore. This is a dummy field used for alignment purposes.

char **date\_in**[4]

Energization Date in format “MYYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

char **date\_out**[4]

De-nergization Date in format “MYYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

float **thermal\_rating**

Thermal rating in Amps.

float **bottleneck\_rating**

Bottleneck rating in Amps.

float **dummy4**

Ignore. This is a dummy field used for alignment purposes.

struct **pf\_itie**

*#include <cflowlib.h>* This structure holds power flow input data for intertie records. It indicates the two areas between which interchange must be scheduled.

### Public Members

char **type**[3]

Two character record type, here “I” for intertie record.

char **area1\_name**[11]

Ten character area1 name.

char **area2\_name**[11]

Ten character area2 name.

float **sched\_export**

Scheduled export power.

struct **pf\_itie\_soln**

*#include <cflowlib.h>* This structure holds power flow solution (output) data for interties which can be.

## Public Members

char **type**[3]

Two character record type.

float **Pexport**

Solution export power.

float **Pcirc**

Solution circulating current.

int **input\_exists**

An integer indicating whether intetie values are internally or externally generated.

0= no input record (internally generated itie)

1= input data is from input record.

struct **pf\_L**

*#include <cfowlib.h>* This structure holds power flow input data for an AC line record (symmetric pi representation).

## Public Members

char **type**[3]

Two character record type “L”.

char **owner**[4]

Three character owner.

char **bus1\_name**[9]

Bus 1 name.

float **bus1\_kv**

Base kV for bus 1.

int **meter**

The line metering point for area tie lines.

1 = bus 1 end.

2 = bus 2 end.

If blank, metering point will be identified (1) by location where line ownership differs from bus ownership or (2) when buses at end of tie line have same ownership, then the bus name 1 will be the metering point.

char **bus2\_name**[9]

Bus 2 name.

float **bus2\_kv**

Base kV for bus 2.

char **ckt\_id**

Circuit identification if more than one parallel branch exists.

int **section**

Section number for making an equivalent for series elements (numeric). Program assembles series elements in numerical order of section numbers (need not be consecutive).

float **total\_rating**

Total ampere rating for all lines.

int **num\_ckts**

Number of parallel circuits represented by this record, for information purposes only.

float **r**

Per unit resistance at base kV and base MVA.

float **x**

Per unit reactance at base kV and base MVA.

float **g**

Per unit conductance at base kV and MVA.

float **b**

Per unit susceptance at base kV and MVA.

float **miles**

Length of the line in miles - information only.

float **dummy1**

Ignore. This is a dummy field used for alignment purposes.

float **dummy2**

Ignore. This is a dummy field used for alignment purposes.

float **dummy3**

Ignore. This is a dummy field used for alignment purposes.

char **descrip**[9]

Information only.

char **date\_in**[4]

Energization Date in format "MYY". Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

char **date\_out**[4]

De-nergization Date in format “MYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

float **thermal\_rating**

Thermal rating in Amps.

float **bottleneck\_rating**

Bottleneck rating in Amps.

float **dummy4**

Ignore. This is a dummy field used for alignment purposes.

struct **pf\_LD**

*#include <cfowlib.h>* This structure holds power flow input data for a two terminal DC line record.

## Public Members

char **type**[3]

Two character record type “LD”.

char **owner**[4]

Three character owner.

char **bus1\_name**[9]

Converter bus 1 name.

float **bus1\_kv**

Base kV for bus 1.

int **meter**

The line metering point for area tie lines.

1 = bus 1 end.

2 = bus 2 end.

If blank, metering point will be identified (1) by location where line ownership differs from bus ownership or (2) when buses at end of tie line have same ownership, then the bus name 1 will be the metering point.

char **bus2\_name**[9]

Converter bus 2 name.

float **bus2\_kv**

Base kV for bus 2.

char **I\_or\_R\_control**

Inverter or rectifier control - Enter 'R' for rectifier control or 'I' for inverter control (point of DC line in which scheduled power is measured).

int **dummy1**

Ignore. This is a dummy field used for alignment purposes.

float **total\_rating**

Total ampere rating for all lines.

int **dummy2**

Ignore. This is a dummy field used for alignment purposes.

float **R**

DC Line resistance, ohms.

float **L\_mh**

DC line inductance, millihenries.

float **C\_uf**

DC line capacitance, microfarads.

float **P\_sched**

Schedule DC power (MW) - Scheduled DC power in megawatts from converter 1 to 2 metered at the end indicated by I or R in I\_or\_R\_control.

float **V\_sched**

Schedule DC line volts (kV) - at rectifier end of DC line.

float **miles**

How many miles long the line is.

float **alpha\_N\_deg**

Initial firing angle in degrees at rectifier.

float **gamma\_0\_deg**

Minimum margin angle in degrees at inverter.

char **dummy3**[9]

Ignore. This is a dummy field used for alignment purposes.

char **dummy4**[4]

Ignore. This is a dummy field used for alignment purposes.

char **dummy5**[4]

Ignore. This is a dummy field used for alignment purposes.

float **thermal\_rating**

Thermal rating in Amps.

float **bottleneck\_rating**

Bottleneck rating in Amps.

float **dummy6**

Ignore. This is a dummy field used for alignment purposes.

struct **pf\_LM**

*#include <cflowlib.h>* This structure holds power flow input data for a multiterminal DC line record.

### Public Members

char **type**[3]

Two character record type “LM”.

char **owner**[4]

Three character owner.

char **bus1\_name**[9]

Converter bus 1 name.

float **bus1\_kv**

Base kV for bus 1.

int **meter**

The line metering point for area tie lines.

1 = bus 1 end.

2 = bus 2 end.

If blank, metering point will be identified (1) by location where line ownership differs from bus ownership or (2) when buses at end of tie line have same ownership, then the bus name 1 will be the metering point.

char **bus2\_name**[9]

Converter bus 2 name.

float **bus2\_kv**

Base kV for bus 2.

char **dummy1**

Ignore. This is a dummy field used for alignment purposes.

int **dummy2**

Ignore. This is a dummy field used for alignment purposes.

float **total\_rating**

Total Current (I) Rating in Amps - Maximum DC line current in ampere.

int **dummy3**

Ignore. This is a dummy field used for alignment purposes.

float **R**

DC Line resistance, ohms.

float **L\_mh**

DC line inductance, millihenries.

float **C\_uf**

DC line capacitance, microfarads.

float **dummy4**

Ignore. This is a dummy field used for alignment purposes.

float **dummy5**

Ignore. This is a dummy field used for alignment purposes.

float **miles**

Length of the line in miles - information only.

float **dummy6**

Ignore. This is a dummy field used for alignment purposes.

float **dummy7**

Ignore. This is a dummy field used for alignment purposes.

char **dummy8**[9]

Ignore. This is a dummy field used for alignment purposes.

char **date\_in**[4]

Energization Date in format “MYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

char **date\_out**[4]

De-nergization Date in format “MYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

float **thermal\_rating**

Thermal rating in Amps.



float **bottleneck\_rating**

Bottleneck rating in Amps.

float **dummy9**

Ignore. This is a dummy field used for alignment purposes.

struct **pf\_qcurve**

*#include <cflowlib.h>* This structure holds power flow input data for reactive power capability.

## Public Members

char **type**[3]

Two character record type - here “QP”, “QM”, or “QN”.

char **PU\_code**[3]

Two character code – PU for per unit or blank for kV values.

char **active**

One character code – “A” for active or “\*” for inactive.

char **bus\_name**[9]

Eight character bus name.

float **bus\_kv**

Base kV of the bus.

float **Pgen0**

Real power levels in MW, for the reactive capability curve. Value is Qmin0 for QN records, Qmax0 for QM cards, and Pgen0 for QP cards, but is always named Pgen0, etc.

float **Pgen1**

Real power levels in MW, for the reactive capability curve.

float **Pgen2**

Real power levels in MW, for the reactive capability curve.

float **Pgen3**

Real power levels in MW, for the reactive capability curve.

float **Pgen4**

Real power levels in MW, for the reactive capability curve.

float **Pgen5**

Real power levels in MW, for the reactive capability curve.

float **Pgen6**

Real power levels in MW, for the reactive capability curve.

float **Pgen7**

Real power levels in MW, for the reactive capability curve.

float **Pgen8**

Real power levels in MW, for the reactive capability curve.

float **Pgen9**

Real power levels in MW, for the reactive capability curve.

struct **pf\_qcurve\_soln**

*#include <cflowlib.h>* This structure holds power flow solution (output) data for reactive power capability curves.

### Public Members

char **type**[3]

Two character record type – here “QP”, “QM”, or “QN”.

float **Pgen**

Real power output level in MW.

float **Qgen**

Reactive power output level in MVAR.

struct **pf\_R**

*#include <cflowlib.h>* This structure holds power flow input data for regulating transformer record. Struct for R, RN, RQ, RV type records.

### Public Members

char **type**[3]

Two character record type “R”.

char **owner**[4]

Three character owner.

char **bus1\_name**[9]

Bus 1 name.

float **bus1\_kv**

Base kV for bus 1.

int **var\_tap\_side**

Variable tap side if T\_max and T\_min cannot orient T\_x.

0 - Low alpha is xed

1 - Bus 1 is variable

2 - Bus 2 is variable

char **bus2\_name**[9]

Bus 2 name.

float **bus2\_kv**

Base kV for bus 2.

char **dummy1**

Ignore. This is a dummy field used for alignment purposes.

int **dummy2**

Ignore. This is a dummy field used for alignment purposes.

float **dummy3**

Ignore. This is a dummy field used for alignment purposes.

int **num\_taps**

Total number of LTC taps. If blank, assumes continuous action.

float **dummy4**

Ignore. This is a dummy field used for alignment purposes.

float **dummy5**

Ignore. This is a dummy field used for alignment purposes.

float **dummy6**

Ignore. This is a dummy field used for alignment purposes.

float **dummy7**

Ignore. This is a dummy field used for alignment purposes.

float **max\_tap**

Maximum kV tap.

float **min\_tap**

Minimum kV taps.

float **dummy8**

Ignore. This is a dummy field used for alignment purposes.

float **dummy9**

Ignore. This is a dummy field used for alignment purposes.

char **rmt\_bus\_name**[9]

Controlled bus name.

char **date\_in**[4]

Energization Date in format “MYYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

char **date\_out**[4]

De-nergization Date in format “MYYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

float **rmt\_bus\_kv**

Controlled bus kV.

float **Qmax**

Maximum reactive (MVAR) flow through the transformer.

float **Qmin**

Minimum reactive (MVAR) flow through the transformer.

struct **pf\_rec**

*#include <cflowlib.h>* Record data.

### Public Members

*input\_data i*

Used to access power flow input data.

*solution\_data s*

Used to access power flow solution (output) data.

struct **pf\_RM**

*#include <cflowlib.h>* This structure holds power flow input data for regulating transformer RM, RP record types.

## Public Members

char **type**[3]

Two character record type “RM”.

char **owner**[4]

Three character owner.

char **bus1\_name**[9]

Bus 1 name.

float **bus1\_kv**

Base kV for bus 1.

int **var\_tap\_side**

Variable tap side if T\_max and T\_min cannot orient T\_x.

0 - Low alpha is xed

1 - Bus 1 is variable

2 - Bus 2 is variable

char **bus2\_name**[9]

Bus 2 name.

float **bus2\_kv**

Base kV for bus 2.

char **dummy1**

Ignore. This is a dummy field used for alignment purposes.

int **dummy2**

Ignore. This is a dummy field used for alignment purposes.

float **dummy3**

Ignore. This is a dummy field used for alignment purposes.

int **num\_taps**

Total number of LTC taps.

float **dummy4**

Ignore. This is a dummy field used for alignment purposes.

float **dummy5**

Ignore. This is a dummy field used for alignment purposes.

float **dummy6**

Ignore. This is a dummy field used for alignment purposes.

float **dummy7**

Ignore. This is a dummy field used for alignment purposes.

float **max\_phase\_shift\_deg**

Maximum angle in degrees.

float **min\_phase\_shift\_deg**

Minimum angle in degrees.

float **dummy8**

Ignore. This is a dummy field used for alignment purposes.

float **dummy9**

Ignore. This is a dummy field used for alignment purposes.

char **rmt\_bus\_name**[9]

Controlled bus name.

char **date\_in**[4]

Energization Date in format “MYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

char **date\_out**[4]

De-nergization Date in format “MYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

float **rmt\_bus\_kv**

Controlled bus kV.

float **Pmax**

Maximum real power (MW) flow through the transformer.

float **Pmin**

Maximum real power (MW) flow through the transformer.

struct **pf\_RZ**

*#include <cflowlib.h>* This structure holds power flow input data for Rapid Adjustment of Network Impedance (RANI) device. It represents a series connected thyristor which changes its series impedance to control power or voltage.

## Public Members

char **type**[3]

Two character record type “RZ”.

char **owner**[4]

Three character owner.

char **bus1\_name**[9]

Bus 1 name.

float **bus1\_kv**

Base kV for bus 1.

int **var\_tap\_side**

Variable tap side if T\_max and T\_min cannot orient T\_x.

0 - Low alpha is xed

1 - Bus 1 is variable

2 - Bus 2 is variable

char **bus2\_name**[9]

Bus 2 name.

float **bus2\_kv**

Base kV for bus 2.

char **ckt\_id**

Circuit identification if more than one parallel transformer exists.

int **section**

Section number for making an equivalent for series elements (numeric). Program assembles series elements in numerical order of section numbers (need not be consecutive).

float **I\_rate**

Rated current.

int **rani\_type**

TYPE 1, 2, or 3

TYPE 1 - Control Pc using Xij TYPE 2 - Control V using Xij TYPE 3 - Control V using Bis

float **Pc\_max**

Pc\_max in MW

float **Pc\_min**

Pc\_min in MW

float **Xij\_max**

Per unit maximum reactance.

float **Xij\_min**

Per unit minimum reactance.

float **Bis\_max**

Per unit minimum reactance.

float **Bis\_min**

Per unit minimum reactance.

float **dummy1**

Ignore. This is a dummy field used for alignment purposes.

float **dummy2**

Ignore. This is a dummy field used for alignment purposes.

char **dummy3**[9]

Ignore. This is a dummy field used for alignment purposes.

char **dummy4**[4]

Ignore. This is a dummy field used for alignment purposes.

char **dummy5**[4]

Ignore. This is a dummy field used for alignment purposes.

float **dummy6**

Ignore. This is a dummy field used for alignment purposes.

float **dummy7**

Ignore. This is a dummy field used for alignment purposes.

float **dummy8**

Ignore. This is a dummy field used for alignment purposes.

struct **pf\_T**

*#include <cfowlib.h>* This structure holds power flow input data for transformer record.



## Public Members

char **type**[3]

Two character record type “T”.

char **owner**[4]

Three character owner.

char **bus1\_name**[9]

Bus 1 name.

float **bus1\_kv**

Base kV for bus 1.

int **meter**

Metering point.

1 = bus 1 end.

2 = bus 2 end.

If blank, metering point will be identified (1) by location where line ownership differs from bus ownership or (2) when buses at end of tie line have same ownership, then the bus name 1 will be the metering point.

char **bus2\_name**[9]

Bus 2 name.

float **bus2\_kv**

Base kV for bus 2.

char **ckt\_id**

Circuit identification if more than one parallel transformer exists.

int **section**

Section number for making an equivalent for series elements (numeric). Program assembles series elements in numerical order of section numbers (need not be consecutive).

float **total\_rating**

Total ampere rating.

int **num\_ckts**

Number of parallel circuits represented by this record, for information purposes only.

float **r**

Per unit resistance at base kV and base MVA.

float **x**

Per unit reactance at base kV and base MVA.

float **g**

Per unit conductance at base kV and MVA.

float **b**

Per unit susceptance at base kV and MVA.

float **tap1**

Fixed bus 1 TAP which describe bus 1 relative to bus 2.

float **tap2**

Fixed bus 2 TAP.

float **dummy1**

Ignore. This is a dummy field used for alignment purposes.

float **dummy2**

Ignore. This is a dummy field used for alignment purposes.

char **dummy3**[9]

Ignore. This is a dummy field used for alignment purposes.

char **date\_in**[4]

Energization Date in format “MYYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

char **date\_out**[4]

De-nergization Date in format “MYYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

float **thermal\_rating**

Thermal rating in Amps.

float **bottleneck\_rating**

Bottleneck rating in Amps.

float **emergency\_rating**

Emergency rating in MVA.

struct **pf\_TP**

*#include <cflowlib.h>* This structure holds power flow input data for phase shifting transformer record.

## Public Members

char **type**[3]

Two character record type “TP”.

char **owner**[4]

Three character owner.

char **bus1\_name**[9]

Bus 1 name.

float **bus1\_kv**

Base kV for bus 1.

int **meter**

The line metering point for area tie lines.

1 = bus 1 end.

2 = bus 2 end.

If blank, metering point will be identified (1) by location where line ownership differs from bus ownership or (2) when buses at end of tie line have same ownership, then the bus name 1 will be the metering point.

char **bus2\_name**[9]

Bus 2 name.

float **bus2\_kv**

Base kV for bus 2.

char **ckt\_id**

Circuit identification if more than one parallel transformer exists.

int **section**

Section number for making an equivalent for series elements (numeric). Program assembles series elements in numerical order of section numbers (need not be consecutive).

float **total\_rating**

Total ampere rating.

int **num\_ckts**

Number of parallel circuits represented by this record, for information purposes only.

float **r**

Per unit resistance at base kV and base MVA.

float **x**

Per unit reactance at base kV and base MVA.

float **g**

Per unit conductance at base kV and MVA.

float **b**

Per unit susceptance at base kV and MVA.

float **phase\_shift\_deg**

Fixed bus 1 TAP or xed phase shift in degrees which describe bus 1 relative to bus 2.

float **tap2**

Fixed bus 2 TAP or blank for xed phase shifter.

float **dummy1**

Ignore. This is a dummy field used for alignment purposes.

float **dummy2**

Ignore. This is a dummy field used for alignment purposes.

char **dummy3**[9]

Ignore. This is a dummy field used for alignment purposes.

char **date\_in**[4]

Energization Date in format “MYYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

char **date\_out**[4]

De-nergization Date in format “MYYY”. Where M = {1,2,3,4,5,6,7,8,9,O,N,D} and YY = last two digits of year.

float **thermal\_rating**

Thermal rating in Amps.

float **bottleneck\_rating**

Bottleneck rating in Amps.

float **emergency\_rating**

Emergency rating in MVA.

struct **pf\_xdata**

*#include <cflowlib.h>* This structure holds power flow input data for switched reactance devices (capacitor and reactor banks).

## Public Members

char **type**[3]

Two character array designating the record type — here “X” for switched reactance record.

char **owner**[4]

Three character array designating an owner.

char **bus\_name**[9]

Eight character bus name.

float **bus\_kv**

Base kV of the BX bus.

char **rmt\_name**[9]

Eight character remote bus name.

float **rmt\_kv**

Remote bus base kV.

int **seg1\_num\_steps**

Number of each reactance value available (integer).

float **seg1\_delta\_mva**

Magnitude of reactance values available.

int **seg2\_num\_steps**

Number of each reactance value available (integer).

float **seg2\_delta\_mva**

Magnitude of reactance values available.

int **seg3\_num\_steps**

Number of each reactance value available (integer).

float **seg3\_delta\_mva**

Magnitude of reactance values available.

int **seg4\_num\_steps**

Number of each reactance value available (integer).

float **seg4\_delta\_mva**

Magnitude of reactance values available.

int **seg5\_num\_steps**

Number of each reactance value available (integer).

float **seg5\_delta\_mva**

Magnitude of reactance values available.

int **seg6\_num\_steps**

Number of each reactance value available (integer).

float **seg6\_delta\_mva**

Magnitude of reactance values available.

int **seg7\_num\_steps**

Number of each reactance value available (integer).

float **seg7\_delta\_mva**

Magnitude of reactance values available.

int **seg8\_num\_steps**

Number of each reactance value available (integer).

float **seg8\_delta\_mva**

Magnitude of reactance values available.

struct **pf\_xdata\_soln**

*#include <cflowlib.h>* This structure holds power flow solution (output) data for switched reactance (capacitor and reactor banks) devices.

### Public Members

char **type**[3]

Two character array designating the record type — here “X” for switched reactance record.

int **seg1\_sch\_units**

Scheduled number of steps.

int **seg1\_used\_units**

Actual number of steps used.

float **seg1\_mvar\_per\_unit**

Actual reactance amounts used per step.

int **seg2\_sch\_units**

Scheduled number of steps.

int **seg2\_used\_units**

Actual number of steps used.

float **seg2\_mvar\_per\_unit**  
Actual reactance amounts used per step.

int **seg3\_sch\_units**  
Scheduled number of steps.

int **seg3\_used\_units**  
Actual number of steps used.

float **seg3\_mvar\_per\_unit**  
Actual reactance amounts used per step.

int **seg4\_sch\_units**  
Scheduled number of steps.

int **seg4\_used\_units**  
Actual number of steps used.

float **seg4\_mvar\_per\_unit**  
Actual reactance amounts used per step.

int **seg5\_sch\_units**  
Scheduled number of steps.

int **seg5\_used\_units**  
Actual number of steps used.

float **seg5\_mvar\_per\_unit**  
Actual reactance amounts used per step.

int **seg6\_sch\_units**  
Scheduled number of steps.

int **seg6\_used\_units**  
Actual number of steps used.

float **seg6\_mvar\_per\_unit**  
Actual reactance amounts used per step.

int **seg7\_sch\_units**  
Scheduled number of steps.

int **seg7\_used\_units**  
Actual number of steps used.

float **seg7\_mvar\_per\_unit**

Actual reactance amounts used per step.

int **seg8\_sch\_units**

Scheduled number of steps.

int **seg8\_used\_units**

Actual number of steps used.

float **seg8\_mvar\_per\_unit**

Actual reactance amounts used per step.

union **solution\_data**

*#include <cflowlib.h>* Solution (output) data union.

### Public Members

*pf\_bus\_AC\_soln* **ACbus**

Use for accessing AC bus solution (output) data.

*pf\_bus\_DC\_soln* **DCbus**

Use for accessing DC bus solution (output) data.

*pf\_branch\_soln* **branch**

Use for accessing branch solution (output) data.

*pf\_area\_soln* **area**

Use for accessing area solution (output) data.

*pf\_itie\_soln* **itie**

Use for accessing intertie solution (output) data.

*pf\_cbus\_soln* **cbus**

Use for accessing continuatino bus solution (output) data.

*pf\_qcurve\_soln* **qcurve**

Use for accessing reactive capability curve solution (output) data.

*pf\_xdata\_soln* **xdata**

Use for accessing switched reactance solution (output) data.

*file* **cflowlib.h**



## Defines

### CFLOW\_IPC\_BUFF\_SIZE

This is the byte size of the in and out buffers used by CLFOW.

## Typedefs

typedef *pf\_R* **pf\_RN**

typedef *pf\_R* **pf\_RQ**

typedef *pf\_R* **pf\_RV**

typedef *pf\_RM* **pf\_RP**

## Enums

### enum **pf\_list\_type**

This is an enumeration of options that you can use in calls to `pf_get_list()`.

*Values:*

#### enumerator **AREA\_LIST**

A list of the different area names in the case. String length 11. Max size 50.

#### enumerator **BUS\_LIST**

A list of the different bus names in the case.

#### enumerator **KV\_LIST**

A list of the different bus kV's in the case. String length 4. Max size 150.

#### enumerator **OWNER\_LIST**

A list of the different owners in the case. String length 4. Max size 450.

#### enumerator **REC\_TYPE\_LIST**

A list of the different record types in the case. String length 3. Max size 50.

#### enumerator **ZONE\_LIST**

A list of the different zones in the case.. String length 3. Max size 150.

## Functions

void **pf\_cflow\_exit**(void)

Close the data link to the powerow engine (ipfsrv).

Call this in to “disconnect” properly from ipfsrv.

### Returns

Has no return; it calls the exit function.

void **pf\_cflow\_init**(int argc, char \*argv[])

Initialize the data link to the powerow engine (ipfsrv).

Establishes a socket connection with the Powerflow process (ipfsrv). Other command line arguments that have been collected by the argv mechanism may be used by the CFLOW program. The command line arguments are “shifted left” such that \*argv[1] contains the first command line argument intended for the CFLOW program and argc is updated to reflect the count of those arguments only.

### Returns

Returns 0 if it is successful; otherwise, it calls the exit function.

int **pf\_cflow\_ipc**(void)

Buffer interface to powerow.

A low-level interface to the interprocess communication that uses two global buffers pf\_cflow\_inbuf and pf\_cflow\_outbuf. This routine is used by most of the other pf functions; however, you can also use it directly. You put valid PCL commands and associated WSCC-formatted data records into pf\_cflow\_outbuf, call pf\_cflow\_ipc, then look for the results in pf\_cflow\_inbuf.

A list and description of the valid PCL commands is in the documentation.

### Returns

Returns 0 if it is successful; otherwise, it calls the exit function.

int **pf\_del\_area**(char \*area)

Delete area by name.

Delete an area along with all buses and all associated branches in the area.

```
error = pf_del_area("ARIZONA");
```

### See also:

pf\_del\_zone

### Parameters

**area** – [in] A string representing an area name.

### Returns

Returns 0 (zero) if it is successful; otherwise, it returns 1.

int **pf\_del\_zone**(char \*zone)

Delete zone by name.

Delete a zone along with all buses and all associated branches in the zone.

```
error = pf_del_zone("NA");
```

**See also:**

pf\_del\_area

**Parameters**

**zone** – [in] A string representing a zone name.

**Returns**

Returns 0 (zero) if it is successful; otherwise, it returns 1.

int **pf\_rename\_area**(char \*oldname, char \*newname)

Rename an area.

Utilizes the powerflow change records of type Z.

**See also:**

pf\_rename\_zone pf\_rename\_bus

**Parameters**

- **oldname** – [in] A string representing an area name to be changed.
- **newname** – [in] A string representing an area name that will become the new name.

**Returns**

Returns 0 (zero) if it is successful; otherwise, it returns 1.

int **pf\_rename\_bus**(char \*oldname, float oldkv, char \*newname, float newkv)

Rename a bus and re-map all associated data to the new name.

Utilizes the powerflow change records of type Z.

**See also:**

pf\_rename\_zone pf\_rename\_area

**Parameters**

- **oldname** – [in] A string representing a bus name to be changed.
- **oldkv** – [in] A floating point number representing the base kV of the bus.
- **newname** – [in] A string representing an area name that will become the new name.
- **newkv** – [in] A floating point number representing the new base kV.

**Returns**

Returns 0 (zero) if it is successful; otherwise, it returns 1.

int **pf\_rename\_zone**(char \*oldname, char \*newname)

Rename a zone.

All zone fields for all records in a zone are updated. If the new zone name already exists, a combined zone results if adjacency permits; otherwise, it is an error.

Utilizes the powerflow change records of type Z.

**Parameters**

- **oldname** – [in] A string representing a zone name to be replaced.
- **newname** – [in] A string representing the new zone name.

**Returns**

Returns 0 (zero) if it is successful; otherwise, it returns 1.

int **pf\_load\_changes**(char \*filename)

Load changes into case.

Passes an ASCII change file name to ipfsrv so that it can read and interpret the file or **filename** contains an “\*\n” followed by change records which are to be processed by ipfsrv. The records must be separated by “\n”.

**Parameters**

**filename** – [in] A string representing a file name, or “\*\n”, followed by valid change records.

**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_load\_netdata**(char \*filename)

Load network data.

Passes a network data file name to the ipfsrv process so that it can read and interpret the network data file. Or contains an “\*\n” followed by bus and branch records which are to be processed by ipfsrv. The records must be separated by “\n”. If a case is currently loaded, it is overwritten and the data is lost. The case loaded is not usable by GUI after CFLOW has completed.

**Parameters**

**filename** – [in] A string representing a file name or an “\*\n” followed by bus and branch records.

**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_load\_oldbase**(char \*filename)

Load a base case.

Passes a base case filename to the ipfsrv process so that it can read and interpret the file as an “oldbase”. If a case is currently loaded, it is overwritten and the data is lost.

**Parameters**

**filename** – [in] A string representing a file name followed by an optional, “rebuild = ON|OFF”.

**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_put\_inrec**(char \*record)

Send WSCC change record to powerow.

Changes, adds, or deletes an input data record for Powerflow. See Record Format section for a description of input record types and rules for adding, changing, and deleting. This function provides a means of inputting records for which there is not a specific function such as factor change (P), although it can be used for inputting any WSCC input record

**Parameters**

**record** – A string containing WSCC formatted data for a Powerflow input data record.

**Returns**

Returns 0 (zero) if it is successful; otherwise, it returns 1.

```
int pf_rec_area(pf_rec *r, char *action)
```

Retrieves, modifies, adds, or deletes area data.

#### Parameters

- **r** – [inout] A pointer to a structure of type *pf\_rec* supplied by the calling routine.
- **action** – [in] A string designating the action to be performed on an area record. See below for the codes and their meanings. Either upper or lower case is acceptable.
  - "F" Retrieves the first area record.
  - "N" Retrieves the next area record. (Area name must be valid.)
  - "G" Retrieves the rest of the area record. (Area name must be valid.)
  - "D" Deletes an area record. (Area name must be valid.)
  - "A" Adds an area record. (All required data fields must be valid. See the IPF Batch User's Guide .)
  - "M" Modifies an area record. (All required data fields must be valid. See the IPF Batch User's Guide .)
  - "E" Eliminates all area records. (area\_rec is ignored and can be set to zero. This code does not delete any zones, buses, etc. It places all zones in area "blank.")
  - "O" Retrieves the solution output data. (The case must be solved and the area name must be valid.)

#### Returns

Returns 0 if it is successful; otherwise, it returns 1.

```
int pf_rec_branch(pf_rec *r, char *action)
```

Retrieves, modifies, adds, or deletes branch records.

To access individual fields of the records use the union member name, for example, using a declaration of *pf\_rec* br; then br.i.branch.tap1 would reference the input tap1 field for a transformer. If you were looking at an E-type line instead, the variable br.i.branch.tap1 would contain the g2 value, but to make it more obvious what you were actually doing, you would probably want to use the union member name br.i.E.g2, instead. All of the structures and unions are declared in cflowlib.h.

#### Parameters

- **r** – [inout] A pointer to a structure of type *pf\_rec*, supplied by the calling routine.
- **action** – [in] A string designating the action to be performed on a branch record. See below for the codes and their meanings. Either upper or lower case is acceptable.
  - "F3" Retrieves the first branch record associated with bus1, bus2, and circuit ID. bus1\_name, bus1\_kV, bus2\_name, bus2\_kV, and ckt\_id must be valid.
  - "N3" Retrieves the next branch record associated with bus1, bus2, and circuit ID. See notes below.
  - "F2" Retrieves the first branch record associated with bus1 and bus2. bus1\_name, bus1\_kV, bus2\_name, bus2\_kV must be valid.
  - "N2" Retrieves the next branch record associated with bus1 and bus2. See notes below.
  - "F1" Retrieves the first branch record associated with bus1. bus1\_name and bus1\_kV must be valid.

"N1" Retrieves the next branch record associated with bus1. See notes below.

"F" Retrieves the first branch record disregarding bus association. All id fields may be null or zero.

"N" Retrieves the next branch record disregarding bus association. See notes below.

"G" Retrieves the rest of the branch record. All id fields must be valid to get a specific record.

"D" Deletes a branch record. All id fields must be valid.

"A" Adds a branch record. All fields appropriate for the branch type must be valid

"M" Modifies a branch record. All fields appropriate for the branch type must be valid.

"O" Retrieves the solution output data.

Notes:

1) with a "wildcard" circuit ID of "" or "\*" and a section code of zero (or blank) - "G" and "F3" are the same as "F2".

2) with a valid (non-"wildcard") circuit ID and a section code of zero (or blank) - "G" is the same as "F3".

3) codes "F", "N", "N1", "N2", and "N3" do not need any data specified in the *pf\_rec*, however the first use of "N", "N1", "N2", and "N3" relies on initialization with a "F", "F1", "F2", "F3", or "G" code on a previous call.

### Returns

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_rec\_bus**(*pf\_rec* \*r, char \*action)

Retrieves, modifies, adds, or deletes bus records.

*pf\_rec* is a union of both input data (i) and solution data (s). To access individual fields of the records use the union member name, for example, using a declaration of *pf\_rec* b; then b.i.ACbus.Pload would contain the MW load for an AC bus, and the smoothing reactance for a DC bus, but to make it more obvious what you were actually doing, you would probably want to use the union member name b.i.DCbus.smooth\_rx\_mh instead, when dealing with a DC bus. If you are retrieving all buses, you can use the ACbus designation for the type, owner, name, kv, and zone fields; the contents of these fields is the same regardless of the bus type.

### Parameters

- **r** – [inout] A pointer to a structure of type *pf\_rec* supplied by the calling routine.
- **action** – [in] A string designating the action to be performed on an intertie record. See below for the codes and their meanings. Either upper or lower case is acceptable.

"F" Retrieves the first bus record.

"N" Retrieves the next bus record. (Name and kV must be valid.)

"G" Retrieves the rest of the bus record. (Name and kV must be valid.)

"D" Deletes a bus record. (Name and kV must be valid.)

"A" Adds a bus record. (All data fields must be valid. See the Record Formats section.)

"M" Modifies a bus record. (All data fields must be valid. See the Record Formats section.)

"O" Retrieves the solution output data. (The case must be solved and name and kV must be valid.)

**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_rec\_cbus**(*pf\_rec* \*r, char \*action)

Retrieves, modifies, adds, or deletes continuation bus (+) data. Note that cbus data is always associated with particular buses.

**Parameters**

- **r** – [inout] A pointer to a structure of type *pf\_rec*, supplied by the calling routine.
- **action** – [in] A string designating the action to be performed on a continuation bus record. See the table below for the codes and their meanings. Either upper or lower case is acceptable.
  - "F1" Retrieves the first continuation bus record associated with a given bus (name, kV).
  - "N1" Retrieves the next cbus record associated with a given bus. (All ID fields must be valid. See the Record Formats section.)
  - "G" Retrieves the rest of the cbus record. (All ID fields must be valid. See the Record Formats section.)
  - "D" Deletes a cbus record. (All ID fields must be valid. See the Record Formats section.)
  - "A" Adds a cbus record. (All data fields must be valid. See the Record Formats section.)
  - "M" Modifies a cbus record. (All data fields must be valid. See the Record Formats section.)
  - "O" Retrieves the output data. (The case must be solved; all ID fields must be valid. See the Record Formats section.)

**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_rec\_comments**(*pf\_comments* \*r, char \*action)

Retrieves or modifies the case name, project title, and case comments.

**Parameters**

- **r** – [in] A pointer to a structure of type *pf\_comments*.
- **action** – [in] A string designating the action to be performed. See below for the codes and their meanings. Either upper or lower case is acceptable.
  - "G" Retrieves the case comments.
  - "M" Modifies the case comments. All data is updated with the contents of the record.

**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_rec\_intie**(*pf\_rec* \*r, char \*action)

Retrieves, adds, modifies, and deletes intertie data.

**Parameters**

- **r** – [inout] A pointer to a structure of type *pf\_rec*, supplied by the calling routine.
- **action** – [in] A string designating the action to be performed on an intertie record. See below for the codes and their meanings. Either upper or lower case is acceptable.
  - "F" Retrieves the first intertie record.
  - "N" Retrieves the next intertie record. (Name1 and Name2 must be valid.)

- "G" Retrieves the rest of the intertie record. (Name1 and Name2 must be valid.)
- "D" Deletes an intertie record. (Name1 and Name2 must be valid.)
- "A" Adds an intertie record. (All data fields must be valid. See the Record Formats section.)
- "M" Modifies an intertie record. (All data fields must be valid. See the Record Formats section.)
- "O" Retrieves the solution output. (The case must be solved and name1 and name2 must be valid.)

**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_rec\_qcurve**(*pf\_rec* \*r, char \*action)

Retrieves, modifies, adds, or deletes reactive power capability curve data.

**Parameters**

- **r** – [inout] A pointer to a calling routine-supplied structure of type *pf\_rec*.
- **action** – [in] A string designating the action to be performed on qcurve record. See below for the codes and their meanings. Either upper or lower case is acceptable.
  - "G" Retrieves the rest of the Q curve records associated with a given bus. (Name and kV must be valid. See the Record Formats section.)
  - "D" Deletes a Q curve record. (Name and kV must be valid. See the Record Formats section.)
  - "M" Modifies a Q curve record. (Valid only for activation or inactivation. See the Record Formats section.)

**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_rec\_xdata**(*pf\_rec* \*r, char \*action)

Retrieves, modifies, and adds switched reactance (X) data.

The delete function is handled by changing the BX bus to another bus type or deleting the BX bus.

**Parameters**

- **r** – [inout] A pointer to a structure of type *pf\_rec*, supplied by the calling routine.
- **action** – [in] A string designating the action to be performed on a switched reactance record. See below for the codes and their meanings. Either upper or lower case is acceptable.
  - "F" Retrieves the first xdata record in a case.
  - "N" Retrieves the next xdata record in a case. (Name and kV must be valid. See the Record Formats section.)
  - "G" Retrieves the xdata record associated with bus\_name and bus\_kV.
  - "A" Adds an xdata record. (All required data must be valid. See the Record Formats section.)
  - "M" Modifies an xdata record. (All required data must be valid. See the Record Formats section.)
  - "O" Retrieves the output data. (The case must be solved; all id fields must be valid. See the Record Formats section.)



**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

```
int pf_rec_a2b(char *net_data, pf_rec *r, char *action)
```

Converts an ASCII record in WSCC format to a *pf\_rec*.

**Parameters**

- **net\_data** – [in] A pointer to a source string of network or output data.
- **r** – [in] A pointer to a structure of type *pf\_rec*.
- **action** – [in] A pointer to a string designating the action to be performed. “I” Converts network data string to binary input data. “O” Converts output data string to binary solution data.

**Returns**

Returns 0 if it is successful; otherwise, it returns -1.

```
int pf_rec_b2a(char *net_data, pf_rec *r, char *action)
```

Converts a *pf\_rec* to an ASCII record in WSCC format.

All data fields in *pf\_rec* must be valid.

**Author**

William D. Rogers

**Date**

1-23-1995

**Parameters**

- **net\_data** – [out] A pointer to a destination string for network data.
- **r** – [in] A pointer to a structure of type *pf\_rec*.
- **action** – [in] A pointer to a string designating the action to be performed on a bus record. See below
  - ”I” Writes network data record.
  - ”D” Writes change record to delete.
  - ”A” Writes change record to add.
  - ”M” Writes change record to modify.
  - ”O” Writes solution record. (The case must be solved.)

**Returns**

Returns 0 if it is successful; otherwise, it returns -1.

```
int pf_save_changes(char *filename)
```

Save a CHANGES file.

Saves to a change file the input data changes you have made to the currently resident base case data.

**Parameters**

**filename** – [in] A string representing a file name.

**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_save\_netdata**(char \*filename, char \*dialect, char \*ratings, int size)

Save network data.

Save network data from a powerflow base case in ASCII format.

**Parameters**

- **filename** – [in] A string representing a file name.
- **dialect** – [in] A string having the following possible values: “BPA”, “WSCC”, “WSCC1”, or “PTI”. These refer to different forms that the output file can take. See ?? for the differences in dialects.
- **ratings** – [in] A string having the following possible values: “EXTENDED”, “NOMINAL”, or “MIN\_EXTENDED”. See ?? for a description of these options for ratings.
- **size** – [in] An integer representing the size of output records - either 80 or 120. The choice of 120 is valid only with the BPA dialect.

**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_save\_newbase**(char \*filename)

Save new base case.

Saves the currently resident base case in its current state to the specified filename.

**Parameters**

**filename** – [in] A string representing a file name

**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_save\_wscs\_stab\_data**(char \*filename, char \*type)

Save WSCC Stability Data.

Saves the power flow data required for input to the WSCC Stability program in either ASCII or binary form depending on the type argument value.

**Parameters**

- **filename** – [in] A string representing a file name.
- **type** – [in] A string representing the type of file format of the saved file. The type values are either “ASCII” or “BINARY”

**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_solution**(void)

Solve the current case.

Causes the powerflow process (ipfsrv) to initiate a solution on the currently resident base case data.

**Returns**

Returns 0 if it is successful; otherwise, it returns -1.

int **pf\_init**(void)

Start up and initialize the powerow engine.

**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_get\_list**(char \*list, int listlen, enum *pf\_list\_type* type, char \*data)

Retrieve list of owners, areas, bus kV's, record types, or zones.

The *list* argument should be a two dimensional character array with the first dimension being the number of areas, and the second dimension being 11 to store area names of up to 10 characters in length. For example, for AREA\_LIST, the array might be declared as

```
char area_names[20][11]
```

if you know that there are not more than 20 area names in the case. *pf\_case\_info* gives information on the number of areas, zones, owners, etc. The *listlen* argument prevents the routine from exceeding the bounds of your array.

#### Parameters

- **list** – [out] A pointer to a two dimensional character array of size [area size][11].
- **listlen** – [in] An integer specifying the maximum number of list elements.
- **type** – [in] The type of list.
- **data** – [in] Any additional data needed for Program Control Language query. Use "" if none.

#### Returns

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_area\_of\_zone**(char \*area, char \*zone)

Finds the name of the area that a zone is in.

#### Parameters

- **area** – [out] A pointer to an array of 11 characters in which the area name is returned.
- **zone** – [in] A string holding a zone name.

#### Returns

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_case\_info**(*pf\_case\_stats* \*r)

Retrieve case info.

Retrieves data from a Powerflow base case and puts it in the info structure.

#### Parameters

- **r** – [out] A pointer to a structure of type *pf\_case\_stats*.

#### Returns

Returns 0 if it is successful; otherwise, it returns 1.

void **pf\_init\_rec**(void \*r, int rtype)

Initialize *pf\_rec* data.

Initializes a data buffer of type *pf\_rec* to blanks and zeros, in order to clear out old data before calling one of the *pf\_rec* routines to store new data in it. Its use is not necessary, but is recommended.

#### Parameters

- **r** – A pointer to a structure of type *pf\_rec*, supplied by the calling routine.
- **rtype** – An enumerated variable defined in ft.h

void **pf\_init\_qcurve**(*pf\_rec* \*r, char \*type, char \*name, float kv)

Initialize all P-Q curve data fields.

Initializes all P-Q curve data fields to 0 except ID fields that are initialized to the values passed as parameters. This function is used to store ID fields for a specific bus before calling *pf\_rec\_qcurve()* to retrieve the generator capability curve values for that bus.

#### Parameters

- **r** – [out] A pointer to a structure of type *pf\_rec* supplied by the calling routine.
- **type** – [in] A string that specifies the record type (must be “QP”).
- **name** – [in] A string that contains the bus name.
- **kv** – [in] A floating point value representing the base kV.

void **pf\_init\_itie**(*pf\_rec* \*r, char \*type, char \*area1, char \*area2)

Initialize all intertie data fields.

Initializes all intertie data fields to 0 except ID fields that are initialized to the values passed as parameters. This function is used to store ID fields for a specific tie line before calling *pf\_rec\_itie()* to retrieve the values for that line.

#### Parameters

- **r** – A pointer to a structure of type *pf\_rec*, supplied by the calling routine.
- **type** – A string that specifies the record type (must be “I”).
- **area1** – A string which contains the area 1 name.
- **area2** – A string which contains the area 2 name.

void **pf\_init\_cbus**(*pf\_rec* \*r, char \*type, char \*owner, char \*name, float kv, char \*year)

Initialize all continuation bus fields.

Initializes all continuation bus data fields to 0 except ID fields which are initialized to the values passed as parameters. This function is used to store ID fields for a specific bus before calling *pf\_rec\_cbus()* to retrieve the continuation record values for that bus.

#### Parameters

- **r** – [out] A pointer to a structure of type *pf\_rec*, supplied by the calling routine.
- **type** – [in] A string that specifies the record type (must be “+”).
- **owner** – [in] A string that contains the owner name.
- **name** – [in] A string that contains the bus name.
- **kv** – [in] A floating point value representing the base kV.
- **year** – [in] A two character string that contains the code year.

void **pf\_init\_bus**(*pf\_rec* \*r, char \*type, char \*name, float kv)

Initialize all bus data fields.

Initializes all bus data fields to 0 except ID fields which are initialized to the values passed as parameters. This function is used to store ID fields for a specific bus before calling *pf\_rec\_bus* to retrieve the values for that bus.

#### Parameters

- **r** – [out] A pointer to a structure of type *pf\_rec*, supplied by the calling routine.

- **type** – [in] A string that specifies the record type (either B or any bus type is legal).
- **name** – [in] A string that contains the bus name.
- **kv** – [in] A floating point value representing the base kV.

void **pf\_init\_branch**(*pf\_rec* \*r, char \*type, char \*name1, float kv1, char \*name2, float kv2, char cid, int sid)  
Initialize all branch data fields.

Initializes all branch data fields to 0 except ID fields which are initialized to the values passed as parameters. This function is used to store ID fields for a specific branch before calling *pf\_rec\_branch* to retrieve the values for that branch.

#### Parameters

- **r** – [out] A pointer to a structure of type *pf\_rec*, supplied by the calling routine.
- **type** – [in] A string which specifies the record type (L, T, E, or specific R-type).
- **name1** – [in] A string that contains the bus 1 name.
- **kv1** – [in] A floating point value representing the base kV for bus 1.
- **name2** – [in] A string that contains the bus 2 name.
- **kv2** – [in] A floating point value representing the base kV for bus 2.
- **cid** – [in] A string that contains the circuit ID. For solution data, '\*' will retrieve the sum of all parallel circuits.
- **sid** – [in] A integer value representing the section ID. For solution data, a value of 0 will retrieve the total equivalent line.

void **pf\_init\_area**(*pf\_rec* \*r, char \*type, char \*name)

Initialize all area data fields.

Initializes all area data fields to 0 except ID fields which are initialized to the values passed as parameters. This function is used to store ID fields for a specific area before calling *pf\_rec\_area* to retrieve the values for that area.

#### Parameters

- **r** – [in] A pointer to a structure of type *pf\_rec*, supplied by the calling routine.
- **type** – [in] A string that specifies the record type (must be A).
- **name** – [in] A string that contains the area name.

int **pf\_bus\_exists**(char \*name, float kv)

See if a bus exists.

#### Author

William D. Rogers

#### Date

8-17-1994

#### Parameters

- **name** – [in] A pointer to a string containing the bus name.
- **kv** – [in] A real value representing the bus base kV.

### Returns

Returns 0 if the bus exists; otherwise, it returns 1.

int **pf\_user\_init\_def**(void)

Initialize User Analysis definitions.

Initializes the user analysis arrays in powerflow (IPF). It should be called prior to other user analysis functions. It sends the command /INITDEF to ipfsrv.

### Author

William D. Rogers

### Date

1-6-1995

### Returns

Returns 0 if it is successful; otherwise, it returns -1.

int **pf\_user\_load\_def**(char \*definitions)

Load User Analysis definitions into IPF.

Loads the user analysis arrays in powerflow (IPF) with the specified symbol definitions. It sends the command

`/LOADDEF`

, followed by a newline (

`'\n'`

) separated list of definitions, to powerflow.

### Author

William D. Rogers

### Date

1-6-1995

### Parameters

**definitions** – [in] A pointer to a string containing User Analysis Define statements.

### Returns

Returns 0 if it is successful; otherwise, it returns -1.

int **pf\_user\_sub\_def**(char \*base)

Substitute User Analysis definitions.

Performs character string substitutions using computed base case quantities upon the tokens defined with the >DEFINE statements within comment records sent to powerflow either through the pf\_user\_load\_def function or a User Analysis file. The return message is available in the global buffer reply\_pf. It sends the command

`/SUBDEF, SOURCE=<name>`

, where name is either “BASE” or “ALTERNATE\_BASE” (“ALT”).

**Author**

William D. Rogers

**Date**

1-6-1995

**Parameters**

**base** – [in] A pointer to a string designating the source base case as “BASE” or “ALTERNATE\_BASE” (may be abbreviated to “ALT”).

**Returns**

Returns 0 if it is successful; otherwise, it returns -1.

int **pf\_user\_report**(char \*filename, char \*output, char action)

Load a User Analysis file for customized analysis listings.

Loads a user analysis file for generating customized analysis listings. The requested report is appended to the output file, which is created if it doesn’t already exist.

**Author**

William D. Rogers

**Date**

1-6-1995

**Parameters**

- **filename** – [in] A character string representing a user analysis file name.
- **output** – [in] A character string representing an output report file name.
- **action** – [in] A string designating the action to be performed.

**Returns**

Returns 0 if it is successful; otherwise, it returns -1.

int **pf\_user\_define**(char \*symbol, char \*id, char \*type)

Define User Analysis Symbol.

Builds a symbol definition and loads it into the user-analysis arrays in powerflow (IPF). It sends a command constructed as follows:

```
/LOADDEF
> DEFINE_TYPE <symbol_type>
LET <symbol_name> = <id_of_computed_quantity>
```

If blanks are part of the quantity id, substitute them with pound signs (#). All data is case insensitive. Symbol names are limited to six characters. Use blanks or commas to separate identity items.

**Author**

William D. Rogers

**Date**

1-30-1995

**Parameters**

- **symbol** – A pointer to a string containing a symbol name.
- **id** – A pointer to a string containing quantity identity.
- **type** – A pointer to a string containing symbol type.

**Returns**

Returns 0 if it is successful; otherwise, it returns -1.

int **pf\_user\_comment**(char \*symbol, char \*suffix, char \*format)

Define User Analysis comment record.

Builds a comment record and loads it into the user-analysis arrays in powerflow (IPF). It sends a command constructed as follows:

```
/LOADDEF  
C <symbol_name><symbol_suffix> = $<symbol_name><symbol_suffix><format>
```

All data is case insensitive, however, the case is preserved. Symbol names are limited to six characters. The suffix is optional and is used for BUS\_INDEX and ZONE\_INDEX data types. Use a null string (“”) if the suffix is not applicable. The format obeys the Fortran convention and can be either floating point (i. e. /F8.3) or text (i. e. /A7). The default is /F6.0. The format string must include the slash (/).

Comment cards constructed by pf\_user\_comment are designed to be processed by either pf\_user\_quantity to retrieve floating point values or pf\_user\_string to retrieve textual information.

**Author**

William D. Rogers

**Date**

1-30-1995

**Parameters**

- **symbol** – [in] A pointer to a string containing a symbol name.
- **suffix** – [in] A pointer to a string containing an index suffix.
- **format** – [in] A pointer to a string containing format code.

**Returns**

Returns 0 if it is successful; otherwise, it returns -1.

int **pf\_user\_quantity**(char \*symbol, char \*suffix, float \*quantity)

Retrieve a User Analysis quantity.

Searches the reply\_pf buffer after a pf\_user\_sub\_def function is called for “<symbol\_name><index\_suffix> = “, where the suffix is optional, and scans in the floating point value that immediately follows. Comment cards can be built with pf\_user\_comment or any other applicable method. The case of the symbol and suffix must match. The suffix is optional and is used for BUS\_INDEX and ZONE\_INDEX data types. Use a null string (“”) for the suffix, if the data type does not use a suffix. The suffix must include the period (i.e. “.VK”).

**Author**

William D. Rogers



**Date**

1-30-1995

**Parameters**

- **symbol** – [in] A pointer to a string containing a symbol name.
- **suffix** – [in] A pointer to a string containing an index suffix.
- **quantity** – [in] A pointer to a float to hold the retrieved quantity.

**Returns**

Returns 0 if it is successful; otherwise, it returns -1.

int **pf\_user\_string**(char \*symbol, int length, char \*info)

Retrieve for User Analysis string.

Searches the reply\_pf buffer after a pf\_user\_sub\_def function is called for "<symbol\_name> = " and scans into info the number of immediately following characters specified by length. Comment cards can be built with pf\_user\_comment or any other applicable method. The case of the symbol name must match. Where applicable include the suffix in the symbol name string.

**Author**

William D. Rogers

**Date**

1-31-1995

**Parameters**

- **symbol** – [in] A pointer to a string containing a symbol name.
- **length** – [in] An integer specifying the number of characters to scan.
- **info** – [in] A pointer to a destination string for the scanned data.

**Returns**

Returns 0 if it is successful; otherwise, it returns -1.

int **pf\_user\_branch**(char \*symbol, *pf\_rec* \*r, char \*type)

Define User Analysis Branch Index.

Builds a symbol definition and corresponding comment card based on the data in a *pf\_rec* structure and the supplied symbol and type and loads them into the user-analysis arrays in powerflow (IPF). It sends a command constructed as follows:

```
/LOADDEF
> DEFINE_TYPE <symbol_type>
LET <symbol_name> = <bus1_name> <bus1_kv>[*] <bus2_name> <bus2_kv>[*]
C <symbol_name> = $<symbol_name>/F15.7
```

Bus names and voltages are derived from the *pf\_rec* branch structure. Blanks in the bus names are replaced by pound signs (#). An asterisk (\*) determines at which terminal the line flow is computed. If the metering point in the branch data is 0 or 1, the first bus is selected, if 2 then the second. Symbol names are limited to six characters and are case insensitive, but retain case for the comment card.

**Author**

William D. Rogers

**Date**

1-31-1995

**Parameters**

- **symbol** – [in] A pointer to a string containing a symbol name.
- **r** – [in] A pointer to a structure of type *pf\_rec*.
- **type** – [in] A pointer to a string indicating the quantity type.

**Returns**

Returns 0 if it is successful; otherwise, it returns -1.

**int pf\_user\_itie**(char \*symbol, *pf\_rec* \*r, char \*type)

Define User Analysis intertie symbol.

Builds a symbol definition and corresponding comment card based on the data in a *pf\_rec* structure and the supplied symbol and type and loads them into the user-analysis arrays in powerflow (IPF). It sends a command constructed as follows:

```
/LOADDEF
> DEFINE_TYPE <symbol_type>
LET <symbol_name> = <area1_name> <area2_name>
C <symbol_name> = $<symbol_name>/F15.7
```

Area names are derived from the *pf\_rec* intertie structure. Blanks in the area names are replaced by pound signs (#). Symbol names are limited to six characters and are case insensitive, but retain case for the comment card.

**Author**

William D. Rogers

**Date**

1-31-1995

**Parameters**

- **symbol** – [in] A pointer to a string containing a symbol name.
  - **r** – [in] A pointer to a structure of type *pf\_rec*.
  - **type** – [in] A pointer to a string indicating the quantity type.
- 'P' = INTERTIE\_P  
'Q' = INTERTIE\_Q  
'S' = INTERTIE\_P\_SCHEDULED

**Returns**

Returns 0 if it is successful; otherwise, it returns -1.

int **pf\_user\_bus**(char \*symbol, *pf\_rec* \*r, char \*suffix)

Define User Analysis Bus Index.

Builds a symbol definition and corresponding comment card based on the data in a *pf\_rec* structure and the supplied symbol and suffix and loads them into the user-analysis arrays in powerflow (IPF). It sends a command constructed as follows :

```
/LOADDEF
> DEFINE_TYPE  BUS_INDEX
LET <symbol_name> = <bus_name> <bus_kv>
C <symbol_name><index_suffix> = $<symbol_name><index_suffix>/F15.7
```

Bus name and voltage is derived from the *pf\_rec* bus structure. Blanks in the bus names are replaced by pound signs (#). Symbol names are limited to six characters and are case insensitive, but retain case for the comment card. The suffix must contain the period (i. e. “.VK”).

#### Author

William D. Rogers

#### Date

1-31-1995

#### Parameters

- **symbol** – [in] A pointer to a string containing a symbol name.
- **r** – [in] A pointer to a structure of type *pf\_rec*.
- **suffix** – [in] A pointer to a string containing the BUS\_INDEX suffix.

#### Returns

Returns 0 if it is successful; otherwise, it returns -1.

int **pf\_plot**(char \*cor\_filename, char \*ps\_filename, char \*options)

Create a plot.

Causes powerflow (IPF) to generate a plot. Difference plots may be made by first loading an reference (alternate) base case with *pf\_load\_refbase*() and providing a difference plot coordinate file. *pf\_plot* sends a command constructed as follows:

```
/plot
<cor_filename>
<ps_filename>
<options>
```

#### Author

William D. Rogers

#### Date

7-6-1995

#### Parameters

- **cor\_filename** – [in] A string representing the name of a coordinate file.

- **ps\_filename** – [in] A string representing the name of the postscript file to be created.
- **options** – [in] An optional string (may be NULL), representing a list of comments and options, separated by newline (“\n”). Each option must begin with an “@” character.

### Returns

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_load\_refbase**(char \*filename)

Load a reference base case.

Passes a base case filename to the ipfsrv process so that it can read and interpret the file as a “reference base” (also referred to as an “alternate

base”). This is done prior to requesting difference plots or comparison (difference) reports. If a reference case is currently loaded, it is overwritten and the data is lost.

### Author

William D. Rogers

### Date

7-7-1995

### Parameters

**filename** – [in] A string representing a file name.

### Returns

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_select\_base**(char base)

Select primary base or reference base records.

Allows the *pf\_rec* functions to access the input and solution data in either the primary base case (OLDBASE) loaded with *pf\_load\_oldbase* or the reference (alternate) base case loaded with *pf\_load\_refbase*. The accessed base case initially defaults to the OLDBASE data.

### Author

William D. Rogers

### Date

7-21-1995

### Parameters

**base** – [in] A character indicating which set of data other commands act upon: ‘O’ OLDBASE data or ‘R’ REFBASE data.

### Returns

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_solve\_area**(char base)

Get the area interchange for an area.

Run the solution to obtain the area interchange amount. Can run for either the old base case or the reference base case.

**Author**

William D. Rogers

**Date**

3-6-1996

**Parameters****base** – [**in**] A character indicating the base case for which to calculate area interchange.

'O' = Old base case 'R' = Reference base case

**Returns**

Returns 0 if it is successful; otherwise, it returns 1.

int **pf\_command**(char \*command)

Run PCL commands.

Passes an ASCII control file name to ipfsrv so that it can read and interpret the file.

**Author**

William D. Rogers

**Date**

5-6-1996

**Parameters****command** – [**in**] A string representing PCL commands [and data].**Returns**

Returns 0 if it is successful; otherwise, it returns -1.

**Variables**char **pf\_cflow\_inbuf**[]char **pf\_cflow\_outbuf**[]char **err\_buf**[]char **reply\_pf**[]int **pf\_cflow\_socket**int **cf\_debug***group* **input\_data**

The following structures are used for input to powerflow. This is the set of CFLOW structures used by the “pf\_rec\_...” functions

### *group* **pf\_rec**

Read, write, and modify powerow records.

The `pf_rec_` functions allow powerflow input (network data) and output (solution) data to be retrieved, as well as allowing input data (network data) to be added, modified, or deleted.

### *group* **pf\_user**

The `pf_user_` functions provide a means of using the User Analysis features of the powerflow.

*dir* `/home/docs/checkouts/readthedocs.org/user_builds/bpa-ipf/checkouts/doc-updates/include`

### *example* **pf\_area\_of\_zone.c**

```
int cnt;
char zones[32][3]; /* array for zone list */

pf_get_list((char *)zones, 10, ZONE_LIST, "");

for (cnt = 0; cnt < 10; ++cnt)
{
    char area_name[11];
    int error;
    error = pf_area_of_zone(area_name, zones[cnt]);
    printf("zone %-5s is in area %-10s\n", zones[cnt], area_name);
}
```

### *example* **pf\_bus\_exists.c**

```
int found;

found = pf_bus_exists(new_name, kv);

if (found == 0)
    printf(" - This node already exists! %s\n", b.i.ACbus.name);
```

### *example* **pf\_case\_info.c**

```
int error;

pf_case_stats ci;

error = pf_case_info (&ci);

if (!error)
{
    fprintf (out, "Number of areas = %d\n", ci.num_areas,
              "Number of zones = %d\n", ci.num_zones,
              "Number of buses = %d\n", ci.num_buses,
              "Number of connections = %d\n", ci.num_circuits, /* Not including_
```

(continues on next page)

(continued from previous page)

```

↪parallels*/
        "Number of lines = %d\n", ci.num_branches);

    if (ci.case_soln_status == SOLVED)
        fprintf(out, "This was a solved case.\n\n");
}

```

*example* **pf\_get\_list.c**

```

int cnt;
char owners[64][4];

pf_get_list((char *)owners, 64, OWNER_LIST);

printf("owners=\n");

for (cnt=0; cnt < 64; ++cnt)
{
    printf("%-5s",owners[cnt]);
}

```

*example* **pf\_init\_functions.c**

```

pf_rec r;

printf("Initialize area record == \n");
pf_init_area(&r, "A", "NORTHWEST");
printf("Type = %s Area Name = %s Scheduled export = %7.1f\n\n", r.i.area.type, r.
↪i.area.name, r.i.area.sched_export);

printf("Initialize branch record == \n");

pf_init_branch(&r, "L", "WESTMESA", 345.0, "FOURCORN", 345.0, "1", 0);

printf("Bus1 Name = %s%5.1f Bus2 Name = %s%5.1f R = %7.1f X = %7.1f\n\n",
    r.i.branch.bus1_name, r.i.branch.bus1_kv, r.i.branch.bus2_name, r.i.branch.bus2_
↪kv, r.i.branch.r, r.i.branch.x);

int error;
pf_rec *b;

/* This function is normally used to find data for a specific bus.
pf_init_bus stores the ID fields in the structure. */
pf_init_bus (&b, "B", "SJUAN G1", 22.0);

/* Then pf_rec_bus retrieves the data. */
error = pf_rec_bus (&b, "G");

```

(continues on next page)

(continued from previous page)

```

/* Gets rest of bus data (input). */
if (!error)
    error = pf_rec_bus (&b, "0"); /* Gets output bus data. */

fprintf (out, "Initialize cbus record == \n");
pf_init_cbus (&r,"+", "PNM", "SAN JUAN", 345.0, " ");
fprintf (out, "Bus Name = %s%5.1f Owner = %s Load = %7.1f\n\n", r.i.cbus.name, r.i.
↪cbus.kv, r.i.cbus.owner, r.i.cbus.Pload);

fprintf (out, "Initialize itie record == \n");
pf_init_itie (&r, "I", "NORTHWEST", "BC=HYDRO");
fprintf (out, "Area 1 = %s Area 2 = %s Scheduled flow = %7.1f\n\n",
    r.i.itie.area1_name, r.i.itie.area2_name, r.i.itie.sched_export);

fprintf (out, "Initialize Q=curve record == \n");
pf_init_bus (&r, "QP", "SJUAN G1", 22.0);
fprintf (out, "Bus Name = %s%5.1f Status code = %s\n\n",
    r.i.qcurve.bus_name, r.i.qcurve.bus_kv, r.i.qcurve.active);

pf_init_rec(r, AREA);
pf_init_rec(r, L_LINE);
pf_init_rec(r, AC_BUS);
pf_init_rec(r, CBUS);
pf_init_rec(r, ITIE);
pf_init_rec(r, QCURVE);

```

*example* **pf\_load\_functions.c**

```

int error;
pf_rec br;

error = pf_load_oldbase ("43bus.bse");
printf("Loaded old base 43bus.bse, status = %d\n\n", error);

if (!error) {
    error = pf_load_changes("43bus.chg");
    printf("Loaded change file 43bus.chg, status = %d\n\n", error);
}

error = pf_load_oldbase ("j98cy94.bse, rebuild = ON");

pf_load_refbase("J98CY94.BSE");
pf_select_base('R');
pf_rec_branch(&br, "F")

```

*example* **pf\_main.c**

```

#include "cflowlib.h"

int main(int argc, char *argv[])

```

(continues on next page)



(continued from previous page)

```
{
    pf_cflow_init(argc, argv);
    printf("pf_load_netdata=%d\n", pf_load_netdata("bench.net"));
    printf("pf_solution=%d\n", pf_solution());
    pf_cflow_exit();
}
```

*example* **pf\_plot.c**

```
pf_load_oldbase("A98CY94.BSE");
pf_load_refbase("J98CY94.BSE");
pf_plot("500BUS_DIF.COR", "A98CY94.PS", "");

system("print/queue=EOHQMS_PS A98CY94.PS");
```

*example* **pf\_put\_inrec.c**

The following program uses pf\_put\_inrec to change an input data record in Powerflow and then outputs a success or failure message to the screen.

```
int error;
char record [130];

/* record needs to contain valid change, add, or delete input data */
error = pf_put_inrec ( record );
if (!error)
    printf ("Successfully changed, added, or deleted input record.\n");
else
    printf ("Invalid record.\n"); }
```

*example* **pf\_rec.c**

```
pf_rec c, a, bus, branch;
int error, status;
char net_data[80];
FILE *out;

printf("Enter branch identifying data: ");
gets(net_data);
pf_rec_a2b(net_data, &b, "I");

pf_rec_bus(&bus, "F");
pf_rec_b2a(net_data, &bus, "I");
printf("%s\n", net_data);

pf_cflow_init( argc, argv );
out = fopen ("ipf_report.txt", "w");

/* Obtain case comments */
error = pf_rec_comments (&c, "G");
fprintf (out, "Current case is: %s Description: %s\n\n", c.case_name, c.case_
```

(continues on next page)

(continued from previous page)

```

↪descrip);

fprintf (out, "%s\n", c.h[0]);
fprintf (out, "%s\n", c.h[1]);
fprintf (out, "%s\n\n", c.h[2]);
fprintf (out, "%s\n", c.c[0]);
fprintf (out, "%s\n", c.c[1]);
fprintf (out, "%s\n\n", c.c[2]);

pf_rec_branch(&branch, "0");

/* Area data */
fprintf (out, "\n***** AREA DATA *****\n\n");
cf_debug = 1;
error = pf_rec_area( &a, "F" ); /* get first area */
cf_debug = 0;

status = pf_rec_area( &a, "0" ); /* get first area output*/

while ( !error && !status )
{
    fprintf (out, "Type Area Name  Slack Bus    NZn  Export    Pgen Pload    Ploss_
↪Pexport Vmax Vmin\n");
    fprintf (out, " %s %s %s%5.1f %d %7.2f %7.1f %7.1f %7.1f %7.1f %6.4f %6.4f\n",
        a.i.area.type, a.i.area.name, a.i.area.sbus_name, a.i.area.sbus_kv, a.i.area.
↪num_zones, a.i.area.sched_export, a.s.area.Pgen, a.s.area.Pload, a.s.area.Ploss,
↪a.s.area.Pexport, a.i.area.max_Vpu, a.i.area.min_Vpu);

    fprintf (out, "Zones in Area: %s %s %s \n\n", a.i.area.zones[0], a.i.area.
↪zones[1], a.i.area.zones[2]);

    error = pf_rec_area( &a, "N" ); /* get next area */
    status = pf_rec_area( &a, "0" ); /* get next area output*/
}

pf_init_itie("I ", areaname1, areaname2);
status = pf_rec_itie (&itie, "G");

for (error = pf_rec_branch(&branch,"f1"); error == 0; error = pf_rec_branch(&branch,
↪"n1"))
{
    printf("      %s, kv= %6.1f,name2= %s, kv2= %6.1f\n",
        branch.i.branch.bus1_name, branch.i.branch.bus1_kv, branch.i.branch.bus2_
↪name, branch.i.branch.bus2_kv);
}

for (error = pf_rec_bus(&bus, "f"); error == 0; error = pf_rec_bus(&bus, "n"))
{
    pf_rec_bus(&r,"o");
    printf("name= %s, kv= %6.1f, vmag=%6.1f, vdeg=%6.1f\n",
        bus.i.ACbus.name, bus.i.ACbus.kv, bus.s.ACbus.Vmag, bus.s.ACbus.Vdeg);
}

```

(continues on next page)

(continued from previous page)

```

/* CBUS DATA */
fprintf (out, "\n***** CBUS DATA *****\n\n");      error = pf_rec_bus( &r, "F"
→);          /* get first bus in case */      status = pf_rec_cbus( &r, "F1" ); /
→* is there a cbus record? */
while (!error) {
    do {
        error = pf_rec_bus( &r, "N" ); /* get next bus in case */
        status = pf_rec_cbus( &r, "F1" ); /* is there a cbus record? */
    } while (status);

    while ( !status ) /* loop on bus with cbus record(s) */ {
        status = pf_rec_cbus( &r, "0" );
        fprintf (out, "Type Own  Bus Name          Pload  Qload  Gshunt  Bshunt
→Pgen  Qgn-mx    Qmin  \n");
        fprintf (out, " %s %s  %s%5.1f %7.1f %7.1f %7.1f %7.1f %7.1f %7.1f\n\n
→",
            r.i.cbus.type, r.i.cbus.owner, r.i.cbus.name, r.i.cbus.kv, r.s.cbus.Pload, r.
→s.cbus.Qload, r.s.cbus.Gshunt, r.s.cbus.Bshunt, r.i.cbus.Pgen , r.i.cbus.Qgen_Qmax, r.
→i.cbus.Qmin);
        status = pf_rec_cbus( &r, "N1" );
    }
}

close(out);

```

*example* **pf\_rename.c**

```

int error;
error = pf_rename_area ("NORTHWEST", "NW AREA");
error = pf_rename_bus ("BELL", 69, "BELL1", 60.6);
error = pf_rename_zone ("7", "N7");

```

*example* **pf\_save.c**

```

error = pf_save_changes ("mychanges.chg");
error = pf_save_netdata ("mychanges.net", "WSCC", "NOMINAL", 80);
error = pf_save_newbase ("mychanges.bse");
error = pf_save_wscs_stab_data ("mychanges.asif", "ASCII");

```

## 2.12 Network Diagrams

### 2.12.1 Overview

IPF has two different network diagram presentations. One is the display you see in the GUI graphics, and the other is the hard copy map. The display is for convenience in accessing system data graphically - it does not represent what the plotted map will look like, nor does it need to be ‘pretty’. The hard copy diagram is designed for reports, documentation, and analysis. It can be generated as a report from the graphic display or produced in a batch environment. This section addresses the hard copy diagram which is designed for reports, documentation, and analysis.

Both presentations use the same coordinate file format. The most important coordinate data, like bus icon and name locations, and line bending points, can be edited graphically from the GUI by moving things around and saving the altered coordinate file. However, you will have to plot out the map in order to see how your changes have affected the hard copy appearance.

The basic diagram shows power system components modeled in a power flow study. The diagram is less detailed than a Powerflow listing, but may have more (or different) information than the graphic display. It shows essential bus and branch solution data. The diagram also shows identification data. For example, it shows date, case identification, program version, and the options used to generate the diagram. The diagram can be enhanced by adding to the coordinate file such items as:

- A legend.
- A border.
- A control block for case identification, signatures, etc.
- A case title heading.
- Selected tie line flows and loss summary.
- An inset showing detail in a selected area.
- Any PostScript language objects.

These items will show up on the map plotted from the GUI, even though many of them cannot be added or edited from the GUI. For details on all the hard copy diagram options and capabilities, and the usage of the `ipfplot` and `ipfbat` programs, see *ipfplot* and *ipfnet*.

### 2.12.2 Input Requirements, Output, and Operation

The Plot function is a set of FORTRAN subroutines within `ipfmain`, which build a dynamic PostScript objects file of references to a static PostScript objects file (`pfmaster.post`). The same routines are called by the Print Plot command issued from the GUI, a command file entered via `ipfbat` or the GUI, and by the batch program `ipfplot`. The dynamic file, which is built by “anding” a coordinate file with powerflow data, defines *which data* will appear on a diagram. The static ASCII PostScript file describes *how data* will appear on a diagram.

When you load a coordinate file, for example `name.cor`, into the GUI, the program copies the contents of the file into `name.tmp`. Any changes you make during your interactive session are reflected in the latter file. They will disappear when you exit IPF unless you Save the coordinate file before exiting.

When you select the Print Plot item on the File menu, IPF uses the intersection of the `name.tmp` file and the currently loaded system data to define which data will appear on the diagram. It then combines this with the file `pfmaster.post`, which describes how data will appear on the diagram. The result is a `name.ps` file, which is sent to a PostScript-capable printer using the Printer Destination string you have selected.

## Input Requirements

The input requirements are:

- A coordinate file, built via the GUI and/or an ASCII text editor.
- A solved system network, either from a base case file or currently loaded in IPF.
- The static PostScript file (`pfmaster.post`) defining how data will be shown on the diagram.

## Output

A dynamic PostScript file, which is built by the Plot program, is appended to the static PostScript file and sent to a PostScript interpreter (printer or computer display) to produce a diagram.

## Plot Program Operation

GUI: Select Print Plot from the File pull-down menu. You can also change the options for the particular plot, by selecting Diagram Options and/or Page Options. These override the options which may be specified in the coordinate file.

ipfbat: See *ipfnet*.

ipfplot: Enter `ipfplot coordinate_file base_file_1 [base_file_2]`. This is a strictly batch process which does not require the GUI. A coordinate file name and one solved base file must be provided. The second base file is required only for difference maps.

However it is invoked, the Plot program determines which information should appear on the diagram by examining the coordinate file. It then searches the Powerflow data for bus, branch, area, and intertie data that are identified in the coordinate file. When a match is found, the Powerflow and coordinate data are combined and formatted into a dynamic PostScript file to activate procedures on the static PostScript file. In addition to bus, branch, transformer, area, and intertie records, all other coordinate file records — options, draw, >define, comment, and PostScript — are processed by the Plot program and formatted to invoke procedures on the static PostScript file.

### 2.12.3 Coordinate File

The coordinate file used by the diagram program is the same file that is generated and used by the GUI display. Since the coordinate file is an ASCII file, it can be generated by any ASCII text editor. The records in that file are described in this section.

The Coordinate file consists of primary coordinate data records and supportive coordinate data records. All coordinates are first quadrant positive Cartesian coordinate values in centimeters. The lower left corner of the diagram is coordinate (0,0). The primary coordinate data records those which specify the bus and branch (bend) locations. These records are:

- Bus Record B
- Line Record L
- Transformer Record T
- Area Record A
- Intertie Record I

The Coordinate file may also include supportive record types to produce a diagram suitable for long term documentation. These records are:

- File Identification Record [ID COORD
- Options Record 0
- Define Record >DEFINE\_TYPE
- Comment Record C
- Draw Record D
- PostScript Record P
- Trailer Record 9 or (\*EOR)

For complete details on all of the coordinate file records, see ??.

### File Identification Record - [ID COORD

The file identification record signifies the beginning of a set of coordinate records. See the table below. Options are set to their default condition when this record is encountered. One such record is required at the beginning of the coordinate file, but it may optionally contain several [ID COORD records and their associated coordinate record sets. Multiple [ID COORD records may be used to insert a control block, legend, or an inset of a detailed section of the diagram.

Table 2.12.1: File Identification Record Format

Column	Format	Description
1-9	A9	[ID COORD
9-90		Not used by Plot program, available for user notes.

### Options Record - 0

Options affect the general appearance of a diagram and determine which Powerflow data will be displayed for each bus and branch on a diagram.

The default option values can be overridden by options specified in the coordinate file. If a coordinate file is currently loaded in the GUI, and the options are changed via the Page Options or Diagram Options menus, these will override (replace) those originally specified in the file. If the file is saved, the new options will be saved in it.

The = and , and blank field delimiters may be used interchangeably. Option names may be upper or lower case. Upper case characters shown in the tables indicate the minimum mandatory characters for identifying the option.

Table 2.12.2: Option Record Format

Column	Format	Description
1-2	A 2	OP - identifies an option record.
3-90	A 88	Free field description of option, see tables below for details.

Here is an example of how to specify options in the coordinate file::

```
OPtions OR=L
Options SScale_factor=0.9,0.9
OP BUs_detail=Bus_name,Powerflow_name
```

**Page Options** These are general appearance options. Several of them are interdependent. They are applied in the following order:

- Orientation
- Scale

- Offset
- Border with transparency flag
- Box for identification
- Case location
- Comment location
- Coordinate file name location
- Size
- Legend location

The option values can be set on the Diagram Options menu.

Table 2.12.3: General Appearance (Page) Options

Option	Description
Size=XX.XX,YY.YY	DEFAULT = 21.59 by 27.94 cm (8.5 by 11 inches).
ORienta- tion=Landscape ORienta- tion=Portrait	DEFAULT=Portrait.
Off- set=XX.XX,YY.YY	Lower left of diagram relative to lower left of page. DEFAULT = 0.0, 0.0.
TRans- parency=Transparent TRans- parency=Opaque	DEFAULT for insets is Opaque. Main diagram is always opaque.
Scale_factor=XX.XX	DEFAULT = 1.0, 1.0.
BOr- der=XX.XX,YY.YY	Locates upper right corner of border. DEFAULT = no border.
BX=XX.UL,YY.UL,XX.LR,YY.LR	Locates identification box. XX.UL and YY.UL locate the upper left corner of the box. If XX.LR and YY.LR are zero (0), the box is positioned in the lower right corner of the diagram. Default coordinates are established for CR (coordinate file), CAsE_name, and COmments. These locations can be overridden. A standard border is drawn as near the edge of the paper as most PostScript printers will allow. DEFAULT = no identification box.
CAsE_name=XX.XX,YY.YY	Locates case name from Powerflow program. DEFAULT = no case name.
COm- ments=XX.XX,YY.YY	Locates comments from user entry and Powerflow program. DEFAULT = no Powerflow comments.
CR=XX.XX,YY.YY	Locates coordinate file name DEFAULT = relative to BX.
LG=XX.XX,YY.YY	Locates upper left corner of legend box. DEFAULT = no legend.

### Diagram Options

The options described in the table below determine which powerflow values will be displayed on a diagram.

Those selections that are ON by DEFAULT may be turned off in one of two ways. Some, such as the bus name selection, may be toggled to the abbreviations on the bus coordinate records, full bus name, or full bus name and base kv. Others, such as generation, may be turned off by preceding the value of interest with NO\_. For example, to not show generation::

```
Option BUs_detail=NO_Generation
or
0 BU=NO_G
```

Orientation		Transparency		Paper Size	
<input checked="" type="radio"/> Portrait		<input type="radio"/> Opaque		Width	<input type="text" value="21.59"/>
<input type="radio"/> Landscape		<input checked="" type="radio"/> Transparent		Height	<input type="text" value="27.94"/>
Border Top Right Corner		Case Name Position		21.6 x 28 cm	
X cm	<input type="text" value="21.59"/>	X cm	<input type="text" value="12.7"/>	<input type="text" value="28 x 43 cm"/>	
Y cm	<input type="text" value="27.94"/>	Y cm	<input type="text" value="4.0"/>		
Comments Position		Offset		Scale Factor	
X cm	<input type="text" value="12.7"/>	X cm	<input type="text" value="0.0"/>	X	<input type="text" value="1.0"/>
Y cm	<input type="text" value="3.7"/>	Y cm	<input type="text" value="0.0"/>	Y	<input type="text" value="1.0"/>
Label Box Coordinates (Centimeters)					
<input type="text" value="12.0"/>		<input type="text" value="4.5"/>		<input type="text"/>	<input type="text"/>
TOP LEFT CORNER		BOTTOM RIGHT CORNER			
Legend Position					
X cm	<input type="text" value="5.0"/>	Y cm	<input type="text" value="0.5"/>		
OK			Close		

Fig. 2.12.1: Page Options Dialog Box



You can also specify on the bus coordinate record whether or not the generation and shunt reactance at this bus should be displayed. See bus coordinate records in Table ?? and Figure ?? for details. The no display indicator on the coordinate records can be overridden with a draw more or always draw option for generators or shunt reactors. For example,:

```
Op BUS_detail=AL_Generators
Op BUS_detail=MO_Generators
Option BUS_detail=AL_Shunt
```

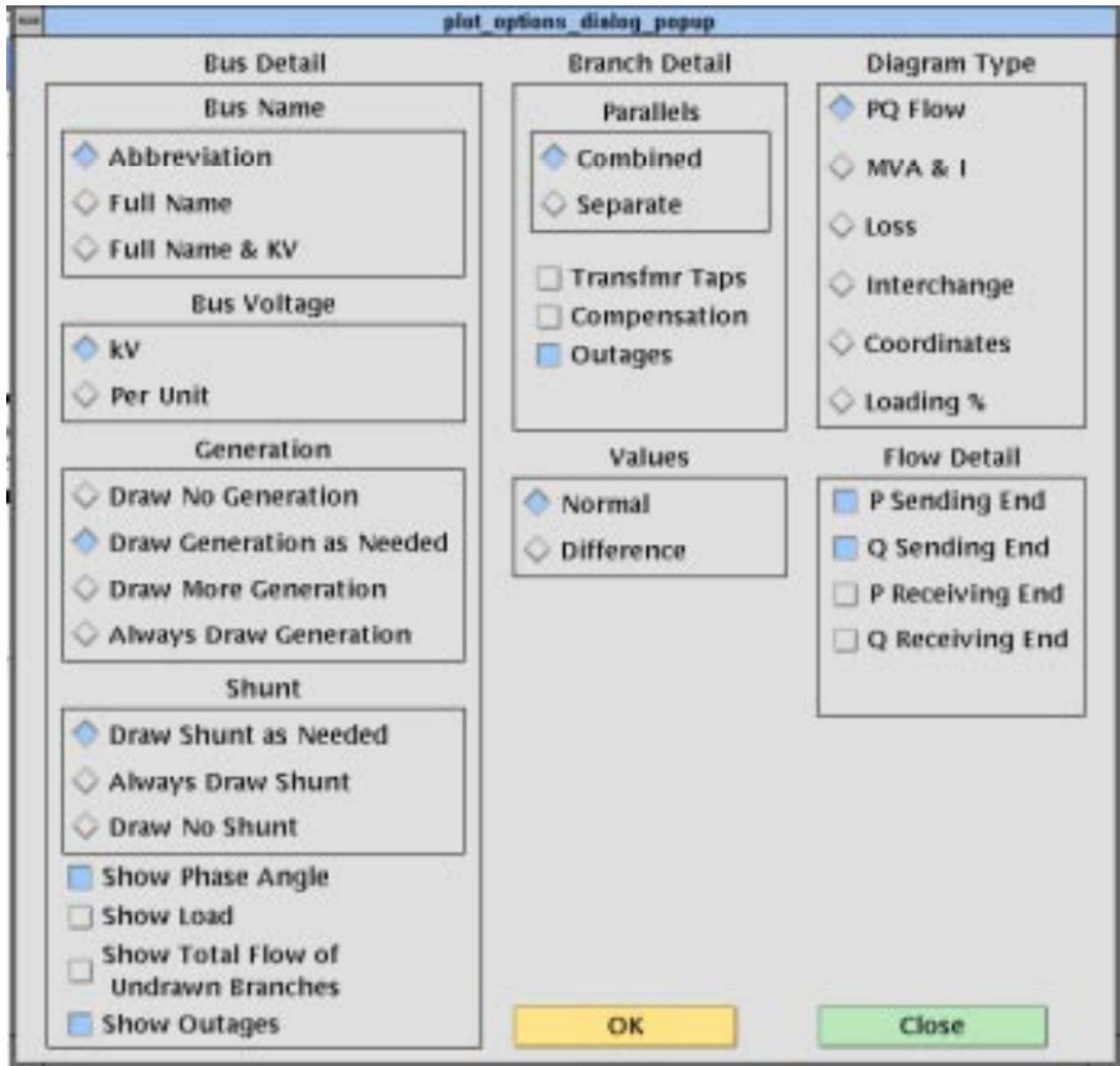


Fig. 2.12.2: GUI Diagram Options

Some of the diagram options are independent; others are mutually exclusive. Look at the GUI menu above to quickly determine which is which. Those with *diamond* buttons are exclusive; those with *square* buttons can be turned on or off in any combination. This applies no matter whether the options are specified from the menu or in the coordinate file.

Table 2.12.4: Powerflow Values Options

Option	Description
DIagram_type=Pq_flow DIagram_type=Mva/I DIagram_type=Loss DIagram_type=Interchange DIagram_type=Coordinates Diagram_type=Loading %	See flow detail; DEFAULT Maximum values P and/or Q set via P_S, Q_S No flow data % current or % MVA rating
Values=Normal Values=Difference	DEFAULT case1 - case2
FLow_detail=P_Sending_end FLOW_detail=Q_Sending_end FLow_detail=P_Receiving FLOW_detail=Q_Receiving	DEFAULT DEFAULT
BUs_detail=Bus_name,Abbreviation BUs_detail=Bus_name,Powerflow_name (and kV) BUs_detail=Voltage,kV BUs_detail=Voltage,Per Unit BUs_detail=Angle BUs_detail=Generation BUs_detail=Shunt BUs_detail=Load BUs_detail=Total_flow of undrawn branches BUs_detail=Outages	DEFAULT DEFAULT DEFAULT DEFAULT DE- FAULT Not yet implemented.
BRanch_detail=Trans_taps BRanch_detail=Compensation BRanch_detail=Parallels,Combined BRanch_detail=Parallels,Separate BRanch_detail=Outages	DEFAULT Not yet implemented.

**File Management** This option is used to ‘include’ a separate file of coordinate data into the current file that you are plotting. CFLOW reports can be incorporated into the diagram by this mechanism. Multiple files may be included, if desired. This option is not presently available from the GUI. That is, if in the GUI you load a coordinate file which includes “Option File” records, these will be ignored. However, there is a means of including a single auxiliary coordinate file in a GUI plot. Under the File pull down menu, select Plot Options, then User Comments. Include a comment which is the name of the file, preceded by an ampersand (&). See section :ref:` for more information.

The ABERDEEN INSET in Figure 5.8 was created with an & record.

The included file should *not* be a complete, independent coordinate file. For example, if it has a [ID COORD record, none of the file will be included. If it contains any options which have already been defined, these will produce warning messages and will have no effect. Options which are not defined in the main file can be defined in the included file; they will be applied to the entire diagram.

The included file must be terminated with a (\*EOR) record, or its last line will be lost.

Table 2.12.5: File Management Option

Option	Description
File=filename	Coordinate data from the specified file will be inserted into the current file. (This feature has not yet been implemented in the GUI.)

## PostScript Records

PostScript, >Define, Comment, and Draw records are processed in the order specified by the user. For example, a PostScript record can be used to change the font for comment records. A comment record may or may not define coordinates. If coordinates are not defined, the comment is printed below the previous comment record. A series of Draw records or a series of PostScript records is often required to accomplish a specific task.

PostScript commands, columns 3 - 82 of the PostScript records, are sent directly to the PostScript file. They have the complete versatility of the PostScript language (PSL). A typical use is to change fonts within a series of comment records or to add simple graphics to a diagram. These records could also be generated, inserted, or edited by a CFLOW program (see also Comment Records, below).

Table 2.12.6: PostScript Record Format

Column	Format	Description
1	A 1	P - identifies a PostScript record.
3-82	A 80	Any valid PostScript command including commands defined in pfmaster.post

### >Define Records

The define records associate solution values to variables that can be manipulated and printed within comment records. Section user-analysis provides an in-depth discussion of this feature. See the table below for the format of >DEFINE records.

Table 2.12.7: Define Record Format

Column	Format	Description
1-7	A 7	>DEFINE — identifies a define record.
1-90	A 90	See user-analysis, /USER- ANALYSIS section, Symbol Definitions.

### Comment Records

Comment records may display simple text or may be used in conjunction with >DEFINE records to display Powerflow values.

They may also be used in conjunction with CFLOW, to create mini-reports on the printed map. Your CFLOW program can edit a coordinate file directly, or create/edit an auxiliary file containing comment records, which is included in the main file(s) with an `OPTION = FILE` record (see also PostScript records, above).

Table 2.12.8: Comment Record Format

Column	Format	Description
1	A 1	C - identifies a comment record.
3-14	2F 6.2	X, Y coordinates of comment. DEFAULT - after previous comment.
15-90	A 76	Comments.

## Draw Records

Draw records are used to draw straight lines such as borders and boxes on a diagram.

Table 2.12.9: Draw Record Format

Column	Format	Description
1	A 1	D - identifies a draw record.
3-14	2F 6.2	X, Y coordinates.
15	I 1	1 = draw or 2 = move to specified coordinates.
16-27	2F 6.2	X, Y coordinates.
28	I 1	1 = draw or 2 = move to specified coordinates.
29-40	2F 6.2	X, Y coordinates.
41	I 1	1 = draw or 2 = move to specified coordinates.
42-53	2F 6.2	X, Y coordinates.
54	I 1	1 = draw or 2 = move to specified coordinates.
55-66	2F 6.2	X, Y coordinates.
67	I 1	1 = draw or 2 = move to specified coordinates.
68-79	2F 6.2	X, Y coordinates.
80	I 1	1 = draw or 2 = move to specified coordinates.

## Bus Coordinate Data

The bus coordinate data describes where and how the Powerflow bus values will be displayed on the diagram. See the table and card below for the format of the bus coordinate data record.

Column	Format	Description
1	A 1	B — Identifies the Bus coordinate record. This record type is created when you place a bus graphically.
2	I 1	Display flag: 0, Blank - Display the bus symbol. 1 - Do not display the bus symbol, but print the name. This is accessed by the Hide Bus and Show Bus buttons in the Bus Coord Edit menu.
3-10	A 8	Bus name to match Powerflow data. Used for identification.
11-14	F 4.0	Bus kv to match Powerflow data. Used for identification.
15-22	A 8	Name abbreviation to print on diagram. This can be edited from the Bus Coord Edit menu.
23	I 1	Print bus voltage relative to bus name: 1 - over name 2 - right of name 3 - below name - this is the GUI-generated default 4 - left of name 5 - do not print the voltage 6 - print the voltage, but no name
24-35	2F 6.2	X, Y of center of bus symbol. This changes whenever you move a bus in the GUI display.
36-47	2F 6.2	X, Y of the lower left corner of the bus name (if other than default). This changes whenever you move a bus name.
48-50	F 3.0	Angle (in degrees) of generator symbol (0 degrees assigns X > 0, Y = 0 position Angle > 0 moves counter clockwise). An angle of 0 is a flag to not display the generator. GUI-generated default is 150 degrees.
51-53	F 3.0	Angle of reactance symbol. An angle of 0 is a flag to not display the reactor. GUI-generated default is 120 degrees.
54-55	A 2	Bus symbol shape identifier: Blank - round symbol - this is the default from GUI. HB - horizontal bar, length = radius x 2 VB - Vertical bar User may add other symbols corresponding to symbols added in master PostScript file.
56-59	F 4.2	Bus symbol radius in centimeters.

[illegible]

### Branch Coordinate Data

The branch coordinate data describes the bending points in a branch and identifies which segment will show the flow and transformer symbol or compensation symbol. See the table and card record below for the format of the branch coordinate data record.

Column 27 requires additional explanation. Several alternative routes may be established for printing parallel circuits separately. The most preferred path is 1, next 2, etc. When the option to display parallel circuits separately is on and there are as many or more routings as circuits, the circuits are shown separately.

Table 2.12.11: Branch Coordinate Data Format

Column	Format	Description
1	A 1	L or T identifies a Line or Transformer. This record type is generated when you place a bending point in a line or transformer.
2		Not used.
3-10	A 8	Bus1 name.
11-14	F 4.0	Bus1 kV.
15-22	A 8	Bus2 name.
23-26	F 4.0	Bus2 kV.
27	I 1	Circuit number for routing parallel circuits separately. Not available from GUI.
28		Not used.
29-30	I 2	Segment for annotation with flow. A negative number means do not show arrow and flow. Not available from GUI.
31-42	2F 6.2	X, Y coordinates for 1st bending point.
43-54	2F 6.2	X, Y coordinates for 2nd bending point.
55-66	2F 6.2	X, Y coordinates for 3rd bending point.
67-78	2F 6.2	X, Y coordinates for 4th bending point.
79-90	2F 6.2	X, Y coordinates for 5th bending point

L or T	BUS NAME 1									BASE KV 1	BUS NAME 2									BASE KV 2	C K T #	A S R E R G O M W N T	X-BEND 1	Y-BEND 1	X-BEND 2	Y-BEND 2	X-BEND 3	Y-BEND 3	X-BEND 4	Y-BEND 4	X-BEND 5	Y-BEND 5				
	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	1	2			3	4	5	6	7	8	9	1	2	3	4	5	6	7
L																																				
T																																				

- ☐ = Required; blanks or zero are unacceptable  
☐ = Optional; blanks or zeros are acceptable  
☒ = Not Applicable

Fig. 2.12.4: Branch Coordinate Data Record



☐ = Required; blanks or zeros are unacceptable  
☐ = Optional; blanks or zeros are acceptable  
☐ = Not Applicable



## Diagram Identification Data

Diagram identification data is retrieved when the user requests the label box (BX) option. Typically, the identification includes:

- Powerflow case name.
- Date of Powerflow run.
- Version of Powerflow program used to create case.
- Description of Powerflow case.
- Time of Powerflow run.

The identification data in the `pfmaster.post` file is of the form shown below. Refer to the `pfmaster.post` file for the identification data in your installation.:

```
%Begin data read by psplot
%[BX WSCC will change the first 7 characters to: %[BPABX
%>DEFINE_TYPE OLDBASE LET cAse=CASE
%>DEFINE_TYPE OLDBASE LET dAte=DATE
%>DEFINE_TYPE OLDBASE LET pFvr=PFVER
%>DEFINE_TYPE OLDBASE LET dEsc=DESC
%P /Helvetica-Bold findfont 10 scalefont setfont
.
.
.
%Option SSpecific Comments=20
%(end)
%[End BPA data block read by psplot --DO NOT REMOVE THIS RECORD
%[WSCCBX WSCC will change the first 4 characters of this record to: %[BX
%! ***** emulated stuff *****
%>DEFINE_TYPE OLDBASE LET tIme=TIME
%P /Helvetica findfont 07 scalefont setfont
.
.
.
%Option SSpecific Comments=03
%(end)
%[End WSCC data block read by psplot --DO NOT REMOVE THIS RECORD
```

## Legend

The legend is in PostScript form and is used when the legend option (LG) is selected. The legend identifies line patterns with kV ranges. The user may examine the procedure `acLine` for a description of the operands passed to that operator.

The legend data in the `pfmaster.post` file is of the form shown below.:

```
/legndId
{
  gsave
  x-legnd y-legnd translate
  0 -4.50 cmtr translate
  newpath
  0 0 moveto 0 4.50 cmtr lineto 2.2 cmtr 4.50 cmtr lineto 2.2 cmtr 0 cmtr lineto
```

(continues on next page)

(continued from previous page)

```

0
closepath stroke
/Helvetica-Bold findfont 10 scalefont setfont
0.4 cmtr 4.00 cmtr moveto
(LEGEND) show
/Helvetica-Bold findfont 08 scalefont setfont
[ 0.10 cmtr 3.25 cmtr 2.10 cmtr 3.25 cmtr ]
() () () 1 0 ( 0 - <100 kv) () () 0 099 acLine
[ 0.10 cmtr 2.50 cmtr 2.10 cmtr 2.50 cmtr ]
() () () 1 0 (100 - <200 kv) () () 0 199 acLine
[ 0.10 cmtr 1.75 cmtr 2.10 cmtr 1.75 cmtr ]
() () () 1 0 (200 - <231 kv) () () 0 230 acLine
[ 0.10 cmtr 1.00 cmtr 2.10 cmtr 1.00 cmtr ]
() () () 1 0 (231 - <500 kv) () () 0 499 acLine
[ 0.10 cmtr 0.25 cmtr 2.10 cmtr 0.25 cmtr ]
() () () 1 0 (500 kv and up) () () 0 500 acLine
grestore
} def

```

## Line Pattern Data

Line pattern data is stored in the array `LinePattern`. The length must be defined in `LinePatternArray`. The put operator stores data in the array element indicated by the indexes 0, 1, 2, 3, etc.

The first two values in the array are the voltage range for the line pattern. Next is an array with a nested array. The nested array is line pattern, and the other value in the offset as required by the operator ``setdash`. The last value in `LinePattern` is the line width.

To be within a voltage range, a voltage must be greater than or equal to the minimum voltage specified and less than the maximum voltage specified.

The line patterns for outages and interchange lines are similar but without the voltage range specification.

The line pattern data in the `pfmaster.post` file is::

```

%***** LINE PATTERN DATA*****
/LinePatternArray 5 def
/LinePattern LinePatternArray array def
/LinePatternCount LinePatternArray 1 sub def
LinePattern 0 [200 231 [ [3 6] 0 ] 1.00 ] put
LinePattern 1 [500 2000 [ [1 1000] 0 ] 1.00 ] put
LinePattern 2 [100 200 [ [5 1] 0 ] 1.00 ] put
LinePattern 3 [231 500 [ [1 5] 0 ] 1.00 ] put
LinePattern 4 [0 99999 [ [1000 1] 0 ] 1.00 ] put

%***** Line Pattern for Outage Line *****

/LinePatternOut [ [ [1 4] 0 ] 1.00 ] def
/curve-offset 8 def

%***** Line Pattern for Interchange Line *****

/LinePatternInt [ [ [1 10000] 0 ] 1.00 ] def

```

## Bus Overvoltage/Undervoltage Range Values

Acceptable bus voltage range data is stored in the array `VoltLim`. The length must be defined in `VoltLimArray`. The `put` operator stores data in the array element indicated by the indexes 0, 1, 2, 3, etc.

The first two values in each element of the array are a voltage range. The second two values are the minimum and maximum acceptable per unit voltages for buses in that voltage range.

To be within a voltage range, a voltage must be greater than or equal to the minimum voltage specified and less than the maximum voltage specified.

```
%***** BUS OVER-VOLTAGE/UNDER-VOLTAGE DATA *****
```

```
/VoltLimArray 5 def /VoltLim VoltLimArray array def /VoltLimCount VoltLimArray 1 sub def VoltLim
0 [230 500 0.95 1.052] put VoltLim 1 [500 2000 1.00 1.100] put VoltLim 2 [115 230 0.95 1.052] put
VoltLim 3 [ 0 115 0.95 1.052] put VoltLim 4 [ 0 9999 1.00 1.000] put
```

## Bus Symbols

The user may add new bus shapes or change the existing ones. The default bus shape is a circle. A vertical bar (VB) and horizontal bar (HB) are also available. See ?? and ?? below. Bus symbols added in the `pfmaster.post` file can then be associated with bus records through the bus shape and scale factor fields on the bus coordinate record.

A typical bus symbol definition in the `pfmaster.post` file is the vertical bar bus symbol::

```
shapeIndex (VB) eq % begin drawing vertical bar
{
  [] 0 setdash
  obj-line-width setlinewidth
  newpath % clear current point so it doesn't show
  -.075 cmtr -1 cmtr bsScale mul moveto
  -.075 cmtr 1 cmtr bsScale mul lineto
  .075 cmtr 1 cmtr bsScale mul lineto
  .075 cmtr -1 cmtr bsScale mul lineto
  closepath
  gsave
  buscolor aload pop setrgbcolor % set background color
  fill
  grestore
  fgcolor aload pop setrgbcolor % set foreground color
  exit
} if % done drawing vertical bar
```

## Area Symbols and Bubble Plots

The user may add new area shapes or change the existing one. The default area shape is a cartouche (round-cornered box). See ?? and ??. Area symbols added in the `pfmaster.post` file can then be associated with area records through the area shape and scale factor fields on the area coordinate record.

The default area symbol definition in the `pfmaster.post` file is the cartouche or bubble::

```
%----- The enclosed code may be customized at the user's pleasure -----

bubFlg 0 ne % do not draw cartouche
{ exit } if
```

(continues on next page)

(continued from previous page)

```

shapeIndex ( ) eq                                % begin drawing cartouche
{
bbSz1 0 eq {/bbSz1 0.80 cmtr def} if % setup default size for bubble
bbSz2 0 eq {/bbSz2 1.50 cmtr def} if % setup default size for bubble
[] 0 setdash
bub-wall-width setlinewidth
newpath                                           % clear current point so it doesn't show
/bbSz3 bbSz2 2 div def
bbSz3 neg 0 bbSz1 90 270 arc
bbSz3 bbSz1 neg lineto
bbSz3 0 bbSz1 270 90 arc
closepath
gsave
bgcolor aload pop setrgbcolor                  % set background color
fill
grestore
fgcolor aload pop setrgbcolor                  % set foreground color

/xTxt xTxt bbSz3 sub                            % calculate new x text coordinates
bbSz1 3 div sub def
/yTxt yTxt 0.5 cmtr add def                    % calculate new y text coordinates

exit
} if                                             % done drawing cartouche

%----- The enclosed code may be customized at the user's pleasure -----

```

## 2.12.5 Diagram Components

Example diagrams are shown in ?? and ??.

The hard copy diagram allows buses, branches, area, and interchange symbols to be intermixed on a diagram. Typically, however, bus/branch and area interchange diagram are drawn separately.

The diagram consists of two types of components:

- Supportive components such as borders, diagram identification, legend, and comments.
- Primary diagram components such as buses, branches, areas, and interchange flows.

### Supportive Diagram Components

The example diagrams illustrate the two different types of default options, one with an identification box in the lower left corner (BPA standard) and the other with case id and headers at the top of the page (WSCC standard).

A border is drawn as close to the edge of the paper as printers allow.

The information below the border identifies the type of diagram, the date and time the diagram was created, the Powerflow program version that created the diagram, the time that the Powerflow case was created, and the name of the coordinate file.

In example 1, the case name, the date the case was created, and the Powerflow program version that created the case are shown on the first line within the box. A 20-character description of the case is on the second line. Comments entered

by the user at the time the diagram was created follow. Note that the last user-entered comment on the bus/branch diagram begins with an ampersand (&). This comment instructs the program to read additional coordinate data from the file `aberdeeninset.cor`.

Blocks of text such as LOSSES on the bus/branch diagram and INTERTIE SCHEDULED ACTUAL on the area interchange diagram are the product of >DEFINE and C records in the coordinate file.

The LG (LeGend) option selects and locates the legend shown on the bus/branch diagram.

In example 2, an IPS-like three line title appears above the map. The first line is composed of the case name, description, date, and program version. The other two lines were entered as headers in the PF ID/Description menu (corresponds to command /HEADER in *Power Flow Control (PFC)*).

### Primary Diagram Components: Bus/Branch Diagrams

The majority of the options previously discussed in the *Options Record - O* section refer to bus/branch diagrams. These options, along with the coordinate file, customize the diagram to the user's specifications. In general, the diagram consists of bus and branch symbol groups and Powerflow solution values. Area and intertie information may appear on the same diagram, but there will be no connection between the two graphs, as is shown below.

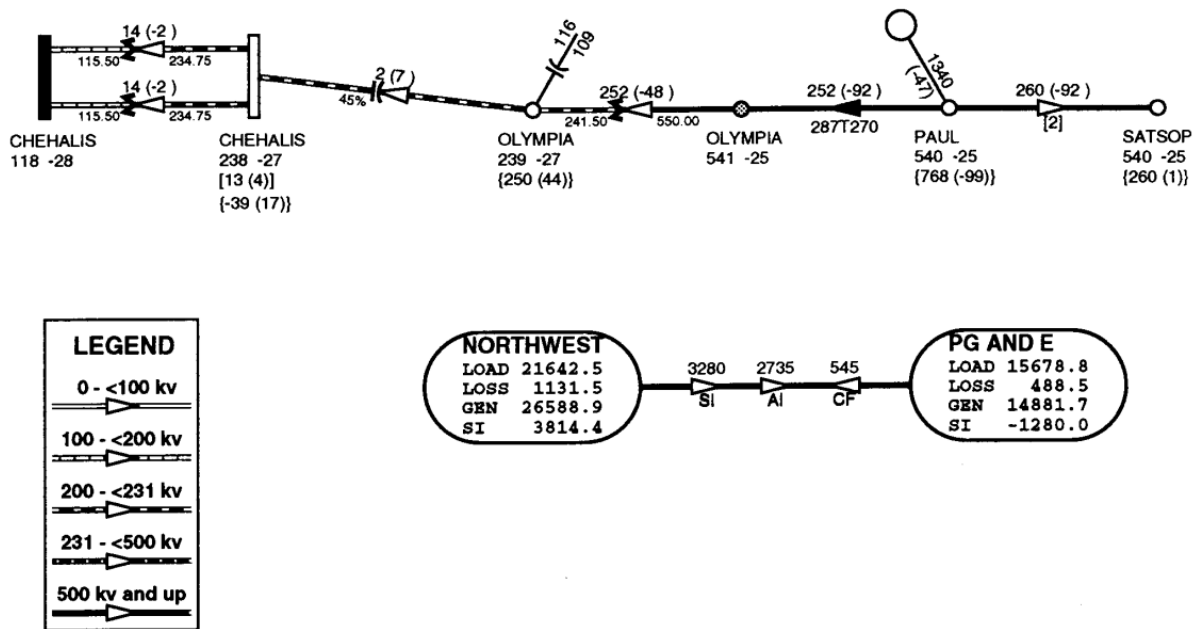


Fig. 2.12.7: Diagram Example

The bus symbol group and values consists of the following:

- Bus symbol — Circle, bar, etc. In the figure above, the bus symbol is fully-shaded (black) — see CHEHALIS 115 — for undervoltage buses and half-shaded (gray) — see OLYMPIA 500 — for overvoltage buses.
- Identification — Powerflow name and kV or abbreviation. The figure below shows buses with the abbreviated form of identification. The CHEHALIS 230 bus is abbreviated CHEHALIS.
- Voltage and angle — Printed above, below, or to the left or right of identification. In the figure above, the CHEHALIS 230 bus voltage is 238 kV and angle -27 degrees.
- Bus load - Printed above, below, or to the left or right of voltage and angle. In the figure above, the load at CHEHALIS 230 is 13 MW and 4 Mvar.

- Total flow on branches to buses not shown on diagram — printed above, below, or to the left or right of load. In the figure above, the net flow at the CHEHALIS 230 bus on branches that are not shown on the diagram is 39 MW into the bus and 17 Mvar out of the bus.
- Generator symbol — Circle connected to bus with a short line segment. Power generation in MW is printed above the line and reactive generation is printed below the line. The figure above shows the generation at PAUL 500 is 1340 MW and -47 Mvar. A close inspection of the data would show the actual generation is on a low voltage bus at the same location as PAUL 500.
- Shunt Reactive — Capacitor or inductor symbol connected to a bus with a short line segment. Maximum reactance available at the bus is printed above the line, and actual reactance used is printed below the line. The figure above shows that 116 Mvar of capacitive reactance is available and 109 Mvar is used at OLYMPIA 230.

The branch symbol group and values consist of the following:

- Line segment symbol - Voltage-coded line segment representing a line or transformer connection between buses. Note the relationship to the legend.
- Arrow symbol - Indicates direction of real power flow through the branch. In the figure above, a solid-shaded arrow (black) indicates the branch is loaded at 100% or more of a rated capacity.
- Overload indicator - Flags circuits that are approaching an overload condition. Branches that are approaching a nominal, thermal, bottleneck, or emergency loading are flagged with an N, T, B, or E under the arrow. The actual current or Mva flow in the line or transformer is before the numeric flag and the rating follows the flag. The figure above shows the PAUL 500 to OLYMPIA 500 line carrying 287 amps and is over 90% of the line's thermal rating of 270 amps. The solid arrow indicates the line is over 100% of the rating. (This is contrived data, of course.)
- Branch flow - Real and reactive power (MW and Mvar) Branch flow is shown above the arrow. Real and reactive sending and receiving end flows are differentiated as follows. MW values are simply shown as numbers. Mvar values are shown within parentheses. Receiving end values are shown within square brackets. Negative values indicate a flow opposite the arrow direction. In the figure above, branch flow is shown above the arrow. Real and reactive sending and receiving end flows are differentiated as follows. MW values are simply shown as numbers. Mvar values are shown within parentheses. Receiving end values are shown within square brackets. Negative values indicate a flow opposite the arrow direction. The power flowing from PAUL 500 to OLYMPIA 500 is 252 MW and -92 Mvar as measured at the sending end.
- Branch flow — MVA and current. An alternative to showing MW and Mvar is to show MVA for transformers and current for lines. The value shown is the maximum for any section of the circuit. If the maximum is at a point other than the sending end, an R is appended to the flow. If a circuit is composed of both line sections and transformer sections, the maximum current for the line section and maximum MVA for the transformer section will be shown. The receiving or sending/receiving end flag is eliminated.

Example: .. image:: ../img/Example\_500\_amp\_650\_mva\_Branch\_Flow.png

This example indicates 500 amps in a line section and 650 MVA in a transformer.

Example: .. image:: ../img/Example\_1000\_amp\_Branch\_Flow.png

- Branch flow - Real and reactive power loss. Real and reactive losses in MW and Mvar are shown above the arrow.  
Example: .. image:: ../img/Example\_Real\_Reactive\_Branch\_Loss.png This example indicates a loss of 1.23 MW and 0.05 Mvar in the branch.
- Parallel circuits - Show number of parallel circuits represented by branch (default). The number of parallel circuits carrying the flow on the diagram is shown in brackets below the line. The figure above shows two parallel circuits between PAUL and SATSOP carrying a total of 260 MW and -92 Mvar.
- Parallel circuits - Show flow on each circuit represented (option). The flow on each of the parallel circuits can be shown separately. The figure above shows two parallel circuits between CHEHALIS 230 and CHEHALIS 115. Each circuit is carrying 14 MW and -2 Mvar.

- Transformer symbol (shown at tip of arrow).
- Transformer taps (shown below transformer). The figure above shows the CHEHALIS 115/230 transformers with taps of 115.50 and 234.75.
- Series compensation symbol (shown at tip of arrow).
- Series compensation value - Percent of line compensation shown below capacitor symbol. The figure shows the CHEHALIS 230 to OLYMPIA 230 line with 45% compensation.

### Primary Diagram Components: Interchange Diagram

The area interchange diagrams are quite simple, showing the area data within the area symbol (bubble), and flow data above the lines connecting areas.

The area symbol and data consists of the following:

- Area symbol — Cartouche (bubble).
- LOAD — Summation of all loads within the specified area.
- LOSS — Summation of all losses within the specified area.
- GEN — Summation of all generation within the specified area.
- SI (Scheduled Interchange) — Export of power from the specified area.

The interchange symbol and data consists of the following:

- Interchange symbol — Line with arrows and values for Scheduled Interchange, Actual Interchange, and Circulating Flow.

The values are shown above the arrows.

The example above shows that 3280 MW was scheduled from NORTHWEST to PG AND E. The actual interchange was 2735 MW. The circulating flow, defined as actual interchange minus scheduled interchange, is 545 MW from PG AND E to NORTHWEST.

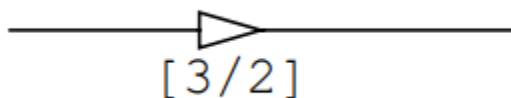
### Primary Diagram Components: Difference Diagram

The format of difference diagrams is very similar to the format of the standard diagram. The standard diagram uses only one case, an active case. The difference diagram uses two cases, an active case and an alternate case.

Displayed values are calculated as active case values minus alternate case values. Arrows indicate the direction of power flow in the active case.

If there are a different number of circuits in the active and reference cases, the number of circuits is in each case shown below the circuit.

Example



This example shows the active case has 3 circuits and the alternate case has 2 circuits in parallel.

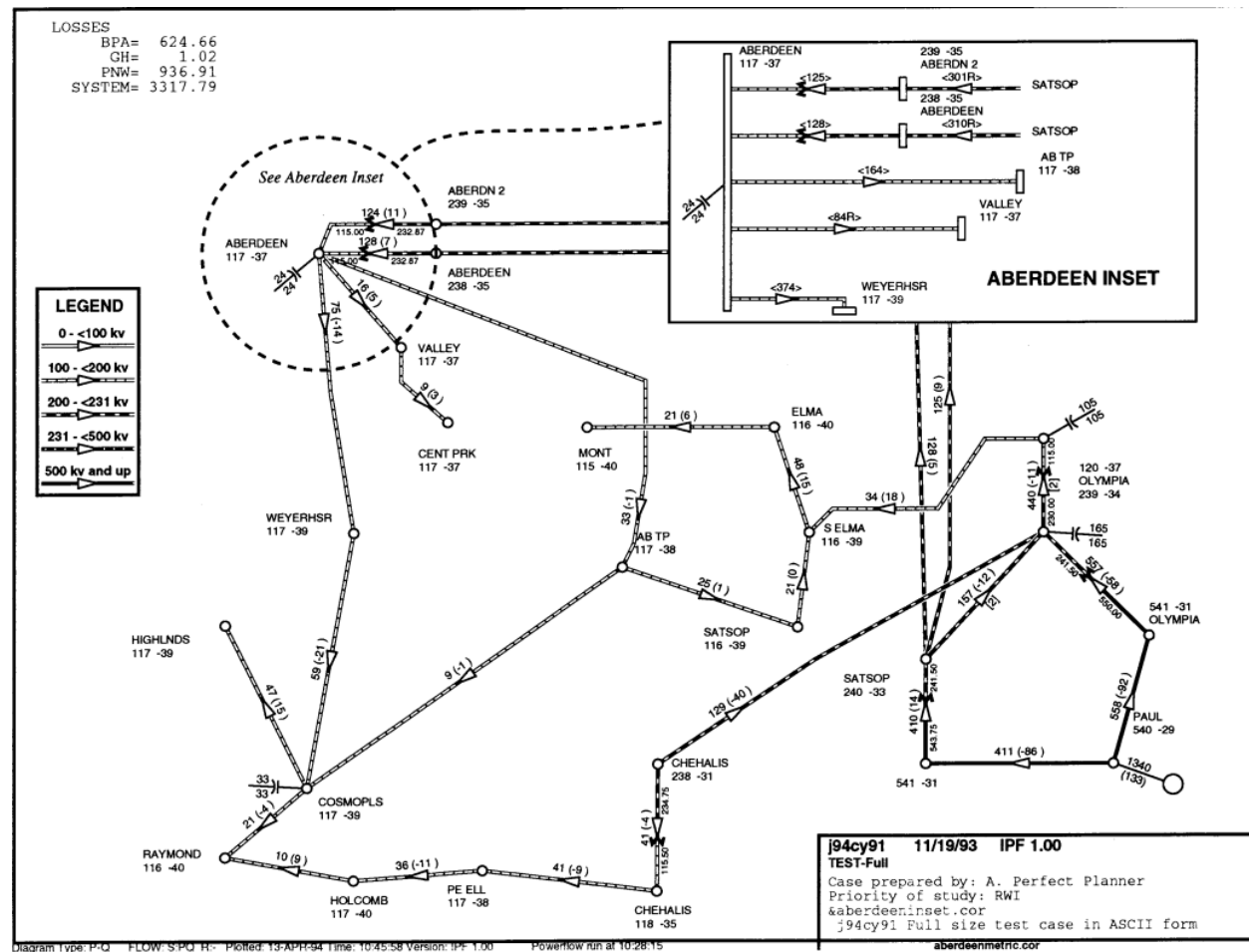


Fig. 2.12.8: Bus Branch Diagram with Inset



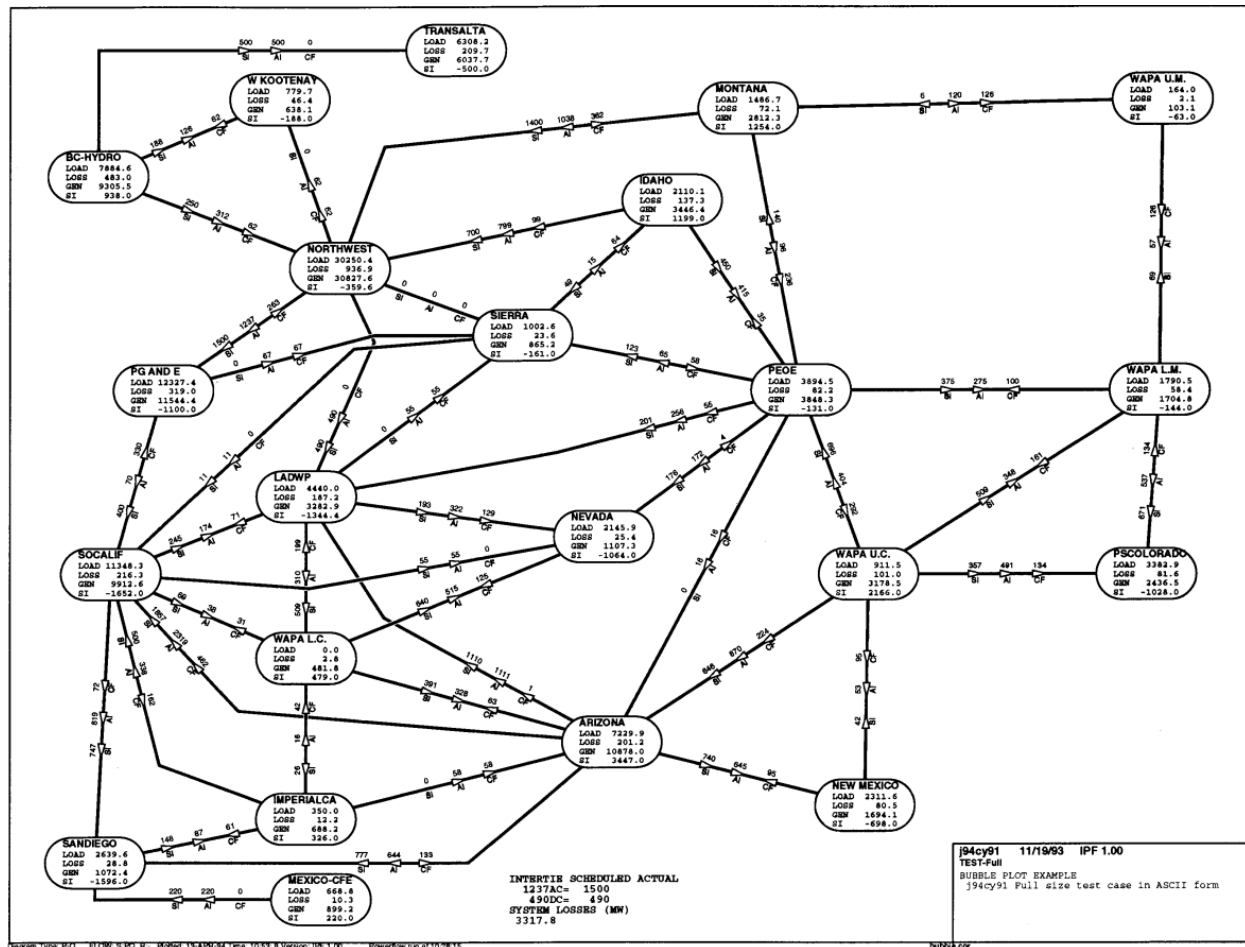


Fig. 2.12.9: Area Interchange Diagram

## 2.13 Calculating Line Impedance

The line impedance subroutine was been extracted from BPA's Electromagnetic Transients Program (EMTP) and modified for computing a balanced pi-equivalent line section. You can find a clone of [BPA's EMTP](#) The line impedance calculation feature is invoked from the branch data dialog. See the screenshot, diagram, and table below. It calculates the pi-equivalent quantities from the conductor type, conductor bundling, and tower geometry. The following applies to all calculations:

- There is continuous transposition of phases.
- Resistivity is calculated at 25 degrees Celsius.
- Earth resistivity is constant at 100 ohm-meters.

The Line Impedance dialog box consists of three panels for data.

- Conductor Values. This is the scrollable list of conductor data, which is usually imported from a line constants data file.
- Edit Conductor. This is the only means to edit data.
- Calculate Impedance. This output panel displays the pi-equivalent data.

### 2.13.1 Description of Conductor Data Fields

Each physical conductor is characterized by a number of items that you need to specify. The following list describes these items:

**UNITS** Allows you to specify either metric or English units.

**BASEKV** The base value for the voltage in kV.

**BASE MVA** The base value for the power in MVA. (Default is 100.0.)

**NAME** Common conductor bird names are hard-coded. In Edit mode, selecting a name from the button menu automatically transfers O.D. `` (outside diameter), ``SKIN, and RESIS data to the pertinent field. If any of these fields are modified, the conductor name becomes "other."

**IPHASE** The phase number (an integer) to which this conductor belongs. Specify as zero for a ground wire (ground is phase number zero by definition). If more than one conductor is specified with the same phase number, this means that those conductors are to be bundled (electrically connected in parallel). Use numbers 1, 2, etc., without any missing (unused) entries, when you number the phases. In general, 1, 2, 3 pertain to circuit1, phases a, b, c, respectively; 4, 5, 6 pertain to circuit2, phases A, B, C, respectively.

**SKIN** A field that is specified with the ratio T/D, where: T is the thickness of the tubular conductor. D is the outside diameter of the tubular conductor. For a solid conductor, use 0.5.

**RESIS** The dc resistance of the conductor in ohms/kilometer if metric or ohms/mile if English.

**DIAM** Outside diameter of the a tubular conductor in centimeters if metric or inches if English.

**HORIZ** Horizontal separation of the center of the conductor from some reference line in meters if metric or feet if English. The location of the horizontal reference line is arbitrary. Distances to the right of the center line are positive, while those to the left are negative.

**VTOWER** Vertical height of the conductor above the ground at the tower in meters if metric or feet if English.

**VMID** Vertical height of the conductor above the ground at mid-span (midway between two towers) in meters if metric or feet if English.

[illegible]

Fig. 2.13.1: Line Impedance Calculation Dialog Box

**Note:** The average height of the line is calculated by the following equation if both the **VTOWER** and **VMID** fields are specified non-zero.

$$VMID + [(VTOWER - VMID)/3] = 2/3xVMID + 1/3xVTOWER$$

If either data field is left blank, the blank field defaults to the other field. In effect, the user specifies the average height.

**SEPAR ALPHA** Leave this blank unless the automatic bundling option is desired.

**FREQ** The frequency,  $f$ , of the line impedance calculation in units of Hertz.

**DIST** The length of the transmission line under consideration in kilometers if metric or miles if English.

**NBUND** Leave this blank unless the following automatic bundling option is desired.

#### AUTOMATIC BUNDLING OPTION

Rather than specifying each conductor of a bundle individually, there is an automatic bundling option that allows a single conductor data to suffice for specifying the entire bundle. This can be used for a “regular” bundle, where by definition all component conductors are identical, and such conductors are also assumed to be uniformly spaced around the circumference of a circle.

Automatic bundling uses the data fields **SEPAR**, **ALPHA**, and **NBUND**, which are otherwise left blank. The “conductor” data then becomes “bundle” data according to the following specifications:

**SEPAR** The separation between adjacent conductors in the bundle in centimeters if metric or inches if English.

**ALPHA** The angular position of the first conductor (or any conductor) of the bundle in units of degrees. Positive angles are measured counter-clockwise as shown in Figure C-1.

**NBUND** The number of conductors that make up the bundle. Or, if you specify by name, separ, and alpha, nbund will be supplied by the program. (For names, see table above under Bundle at the 500 kV lines.)

## 2.13.2 Output Values

Considerable data is required to compute the pi-equivalent for a single segment. The line impedance dialog is most effective when importing files containing line impedance data.

A typical file contains ASCII data in free-field format. The file types are .lcd for “line conductor data.” The following is an example.

```
/GET_DATA, TYPE = LINE_IMPEDANCE_CALCULATION
UNITS = < ENGLISH | METRIC >, DISTANCE = < miles | km >
BASEKV = <basekv>, BASEMVA = <basemva>, FREQUENCY = <freq>
CONDUCTOR = 1 .3636 .05215 1.602 -20.75 50. 50. 0.0 0.0 0
CONDUCTOR = 1 .3636 .05215 1.602 -19.25 50. 50. 0.0 0.0 0
CONDUCTOR = 2 .3636 .05215 1.602 -0.75 77.5 77.5 0.0 0.0 0
CONDUCTOR = 2 .3636 .05215 1.602 0.75 77.5 77.5 0.0 0.0 0
CONDUCTOR = 3 .3636 .05215 1.602 19.25 50. 50. 0.0 0.0 0
CONDUCTOR = 3 .3636 .05215 1.602 20.75 50. 50. 0.0 0.0 0
CONDUCTOR = 0 .5 2.61 0.386 -12.9 98.5 98.5 0.0 0.0 0
CONDUCTOR = 0 .5 2.61 0.386 12.9 98.5 98.5 0.0 0.0 0
```

The usual scenario involves the following steps.

1. Import a relevant .lcd file using the Use Saved button.

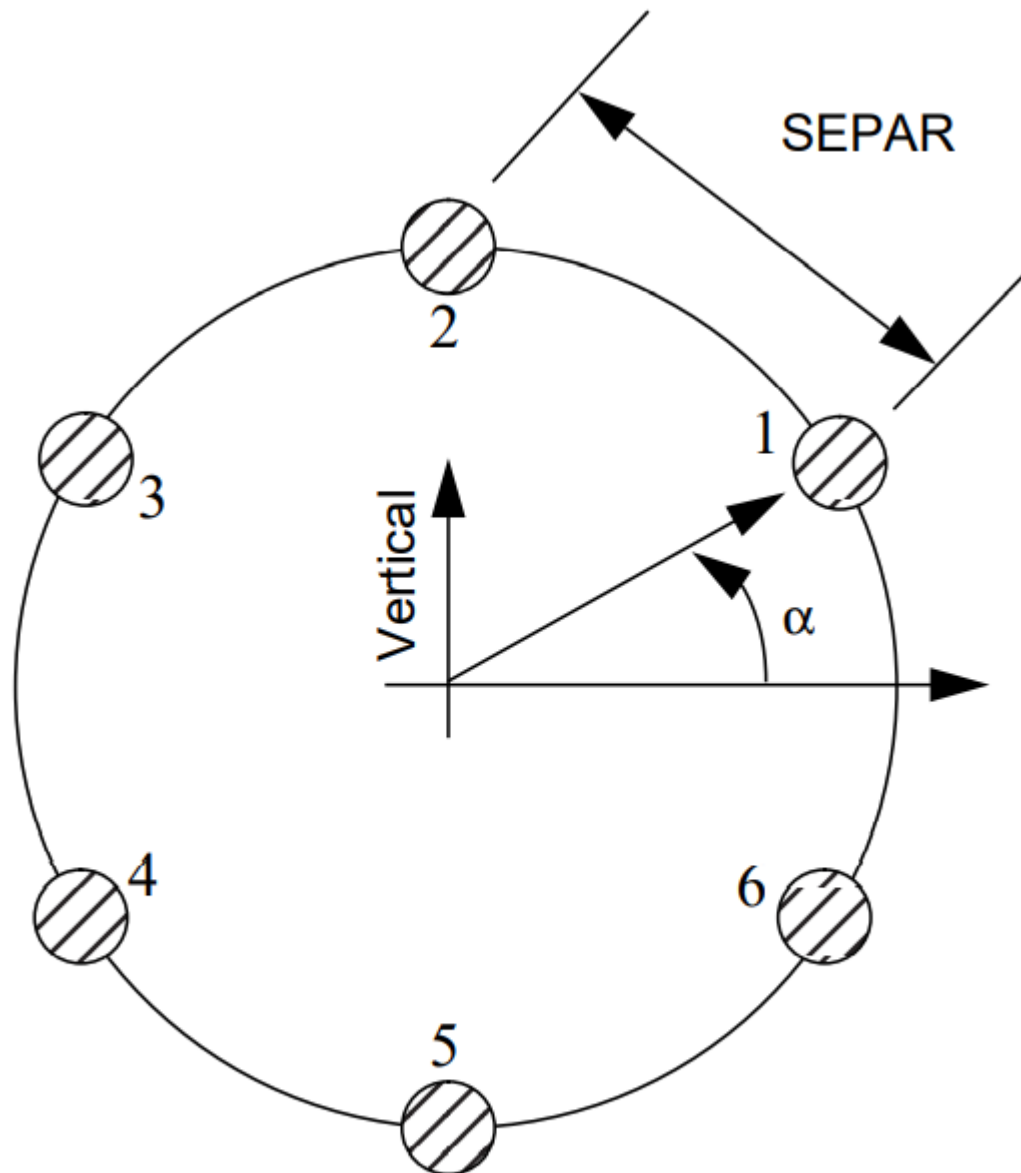


Fig. 2.13.2: A Six Bundle Conductor

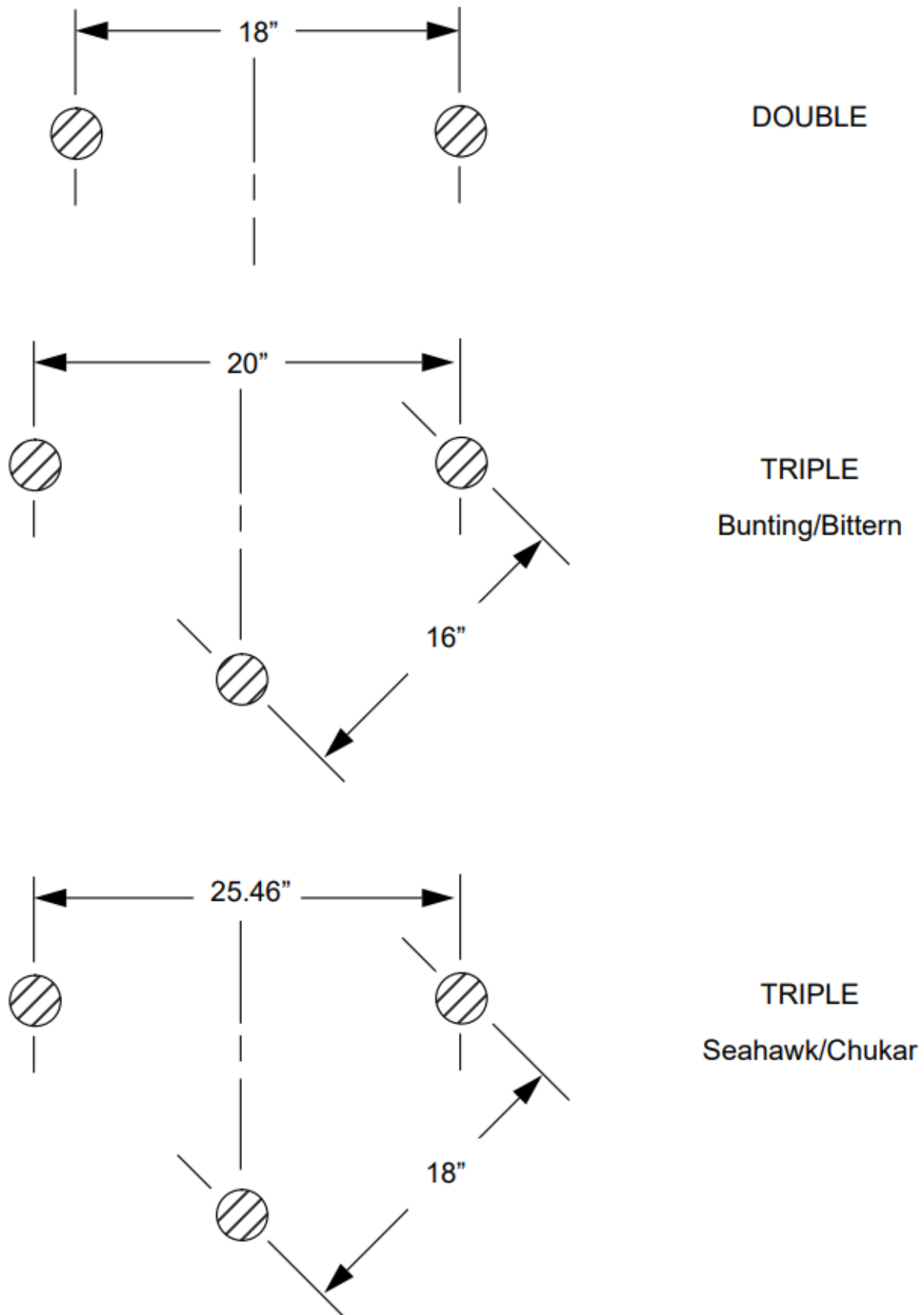


Fig. 2.13.3: BPA Conductor Bundles

2. Edit the file to reflect the geometric and physical attributes of the line under consideration.
3. Calculate the pi-equivalent data. Steps 2 and 3 may be iterated as long as necessary to eliminate errors.
4. Export the modified .lcd file using the Save Values button.
5. If the results are acceptable, automatically transfer the positive sequence values back to the line dialog by pressing the OK button. (The zero sequence values are for inspection only.)
6. If the results are not acceptable, press the Close button. No values are transmitted back to the line dialog by a Close action.

### 2.13.3 Calculating the Impedance

After you have supplied all the necessary values for the line impedance calculations, you can click on the Calculate Impedance button in the Line Impedance Calculation dialog box. The program fills in six numbers in per unit values having the following meanings:

$R/Z1$  transfer Real part of the positive sequence transfer impedance per unit.

$X/Z1$  transfer Imaginary part of the positive sequence transfer impedance per unit.

$R/Z1$  shunt Half value of the real part of the positive sequence shunt admittance per unit.

$X/Z1$  shunt Half value of the imaginary part of the positive sequence shunt admittance per unit.

$R/Z0$  transfer Real part of the zero sequence transfer impedance per unit.

$X/Z0$  transfer Imaginary part of the zero sequence transfer impedance per unit.

## 2.14 DC Line Modeling

The following describes the data representation of a DC line as used in IPF. A typical configuration of the network is shown.

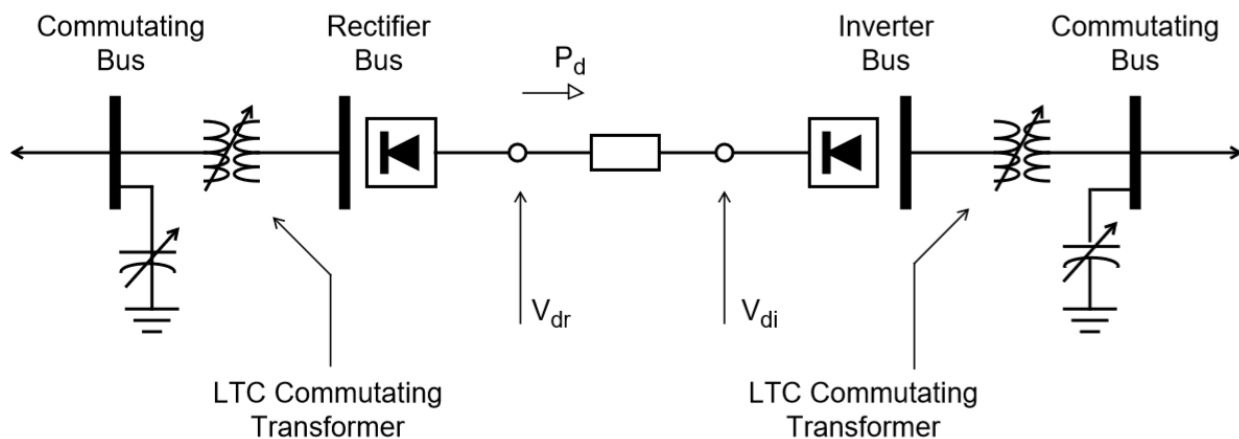


Fig. 2.14.1: Two Pole, Two Terminal DC Line Conguration

Each DC line requires two DC buses, a single DC line and two commutating buses, each of which is connected to the DC bus with a commutating transformer.

Representation of the commutating transformer conforms to the ac network; one transformer is required for each bridge. They are parallel connected in the ac network, necessitating either separate transformer data records or an equivalent

transformer which has the correct parallel impedance and MVA rating. As shown in the diagram, the commutating transformers must be LTC, specifically type R, subtype blank, controlling the DC bus, and oriented such that the tap range encompasses the commutating bus base kV. The magnitude of the DC bus voltage to be controlled is automatically determined from the DC line parameters.

Assuming nominal taps, the DC no-load voltage  $V_{do}$  available from each converter bridge is approximated by the formula.

$$V_{do} = 1.35 * E_v * \cos(\alpha)$$

where  $\alpha$  is the firing angle at the rectifier and the extinction angle at the inverter.  $V_{do}$  is the no-load direct voltage and  $E_v$  is the effective voltage at the DC bus. The actual voltage is a chopped sinusoid.

The full load direct voltage will be less. Normal representation of the DC line usually models the positive and negative poles as separate but parallel DC lines. This scheme permits asymmetrical operation of the DC line. For example, one bridge may temporarily be out of operation. In this case, one pole may have three bridges while the other has only two.

With this scheme, the model may be as shown as follows.

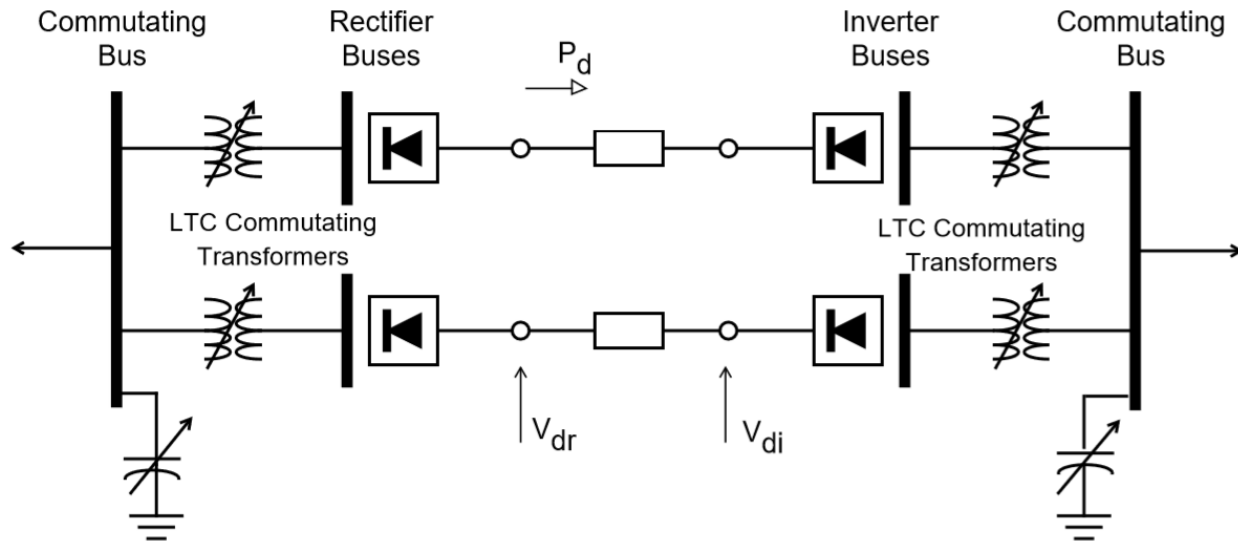


Fig. 2.14.2: Single Pole, Two Terminal DC Line

In the former scheme the dc line quantities are pole-to-pole. In the latter scheme, the quantities are pole-to-neutral. However, the direct voltage must always be positive on the line data record.

When using the pole-to-neutral scheme, the direct voltage of each pole must correspond to the number of bridges in operation. It is recommended that the allocation of the total dc power to each pole should result in no net neutral current, that is, the current in each pole must be equal but opposite.

An example will clarify this. Assume that the normal direct voltages is 800 kV pole-to-pole with six bridges in operation, or 133 kV/bridge. It is desired to find the voltage and power in each pole required to transport 600 MW with one negative bridge out of operation. This situation is shown as follows



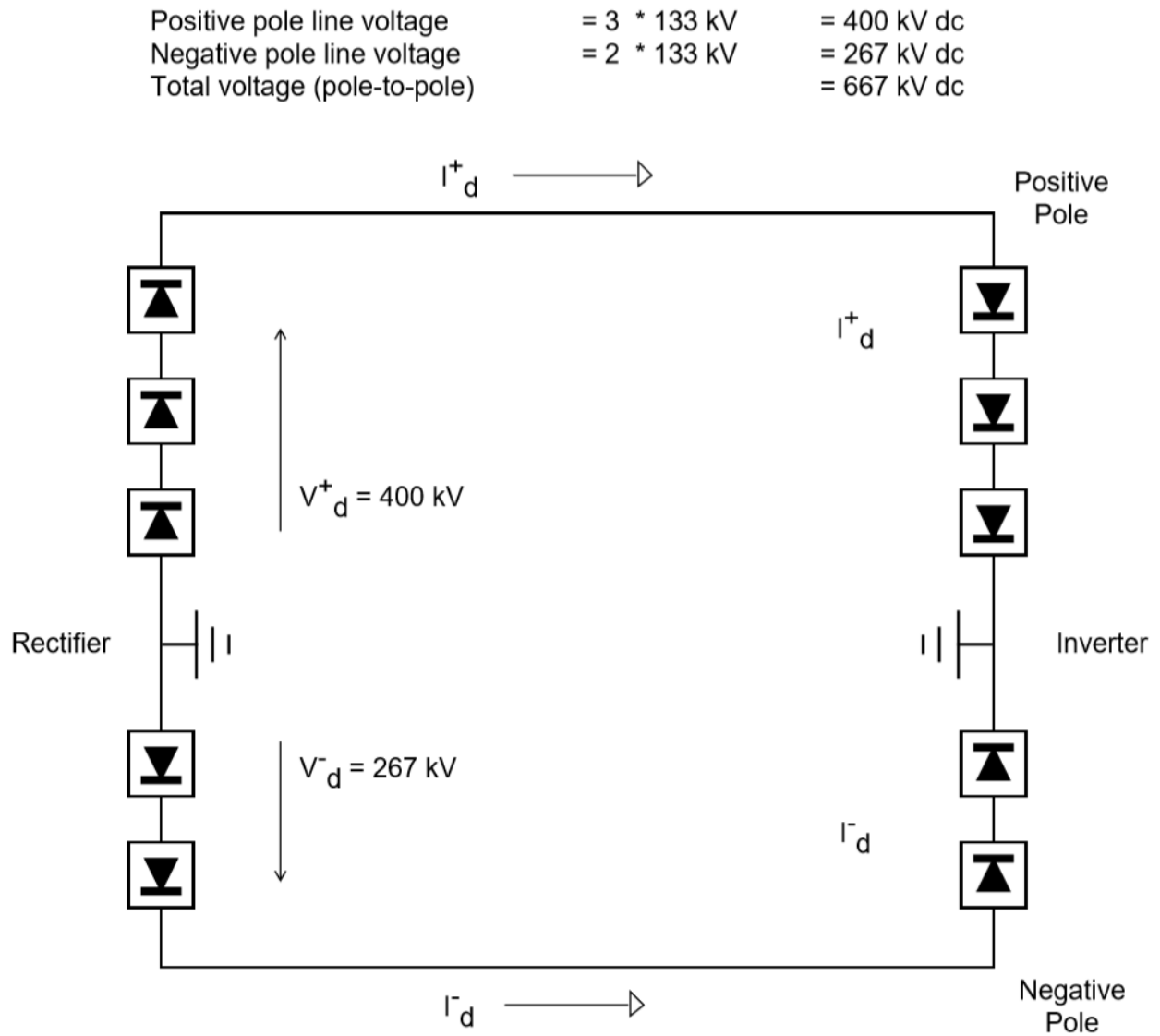


Fig. 2.14.3: One Negative Bridge Out of Operation

For zero neutral current,  $I_d^+ = I_d^-$  (with positive values assigned in the direction shown)

$$I_d = \frac{p_{dc}}{V_{dc \text{ pole-to-pole}}} = \frac{600MW * 1000kV/MW}{667kV} = 900amps$$

$$P_{dc}^+ = V_{dv}^+ * I_{dc}^+ = 400kV * 900amps$$

$$= 360,000kW = 360MW$$

$$P_{dc}^- = V_{dv}^- * I_{dc}^- = 267kV * 900amps$$

$$= 240,000kW = 240MW$$

Assuming the DC line was modeled for six-bridge operation, the only changes required to take a bridge out of service would be modification of the following quantities:

- Positive and negative pole DC line power on the LD record.
- Negative pole DC line voltage on the LD record.
- Delete one of the parallel commutating transformers at each end of the negative pole DC line.
- Change the number of bridges for both converter buses.

## 2.15 Network Reduction

This section gives a detailed description of the network reduction method and solution options.

### 2.15.1 Method of Reduction

This program reduces a large sparse network into a smaller equivalent network by Gaussian decomposition. The original network is linearized about the operating point and is expressed by the current equation:

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \quad (2.15.1)$$

where:  $I_1$  is the complex current net injection matrix,  $Y$  the complex nodal admittance matrix, and  $V$  the complex nodal voltage matrix. The subscripts 1 and 2 pertain to the eliminated and retained portions of the network, respectively. For simplicity, the matrix is reordered as shown. We may separate the partitioned matrix equations in Eq.2.15.1.

$$[I_1] = [Y_{11}] [V_1] + [Y_{12}] [V_2] \quad (2.15.2)$$

$$[I_2] = [Y_{21}] [V_1] + [Y_{22}] [V_2] \quad (2.15.3)$$

We may solve for  $[V_1]$  from Eq.2.15.2.

$$[V_1] = [Y_{11}]^{-1} [I_1] - [Y_{11}]^{-1} [Y_{12}] [V_2]$$

and then substitute the expression into Eq.2.15.3.

$$[I_2] = [Y_{21}] \left[ [Y_{11}]^{-1} [I_1] - [Y_{11}]^{-1} [Y_{12}] [V_2] \right] + [Y_{22}] [V_2]$$

Rearranging terms, we have

$$[I_2] - [Y_{21}] [Y_{11}]^{-1} [I_1] = \left[ [Y_{22}] - [Y_{21}] [Y_{11}]^{-1} [Y_{12}] \right] [V_2]$$

or

$$[I_2]_{eq} = [Y_{22}]_{eq} [V_2] \quad (2.15.4)$$

where:

$$[I_2]_{eq} = [I_2] - [Y_{21}] [Y_{11}]^{-1} [I_1] \quad (2.15.5)$$

$$[Y_{22}]_{eq} = [Y_{22}] - [Y_{21}] [Y_{11}]^{-1} [Y_{12}] \quad (2.15.6)$$

The matrix operations in Eq.2.15.5 and Eq.2.15.6 have the following interpretation:

Equivalent injection = original injection + distributed injection

Equivalent admittance = Original admittance + distributed admittance + equivalent branches

Equations Eq.2.15.5 and Eq.2.15.6 have interesting topological interpretations. Some notation is necessary so the following definitions apply:

Envelope node

A retained node with at least one adjacent node in the eliminated system.

Internal node

A retained node with all adjacent nodes in the retained network.

Equivalent branch

A fictitious branch between two envelope nodes which effectively represents the reduced network as seen from those nodes.

Inherent in the reduction is the connectedness of the network and the preserved identity of the current equations. An eliminated node is connected (not necessarily directly) to several envelope nodes. Any injected current on that node becomes branch current in the eliminated system. After deducting losses, it finally reaches the envelope nodes and is reconverted into an equivalent injection. The distribution of eliminated injections is determined by the admittance of the eliminated system. By superposition, the injections from all eliminated nodes are distributed to the envelope nodes. This reflects the second term on the right side of equation Eq.2.15.5. This does not imply, however, that the equivalent injections are identical to the branch currents and could also be obtained by arbitrary cutting of the network. The reasons will be explained in examples to follow.

The equivalent branches introduced between the envelope nodes after reduction reflect the admittance seen from the envelope nodes and into the eliminated network and finally to other envelope nodes. It is similar to the delta branches introduced in a wye-delta conversion. The value of the equivalent branches is found in the second term in the right side of equation Eq.2.15.6.

The branch data is originally submitted in the form of equivalent pi's which are used to construct the nodal admittance matrix Y. In general, the reverse process is not unique. The figure below illustrates the form of the equivalent pi-branches obtained.

In the figure below, a nine-node network is reduced. The distributed injections and equivalent branches introduced are emphasized.

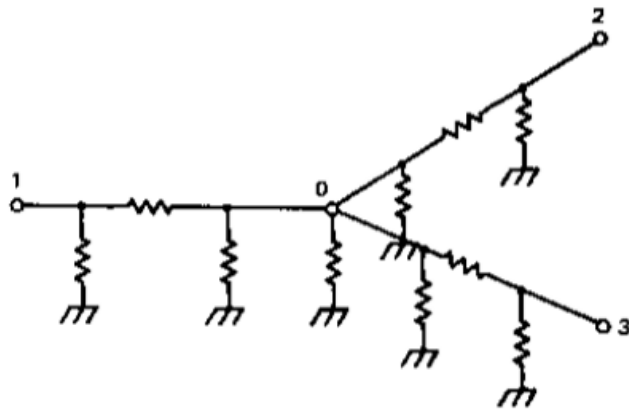
Three options are available for disposal of the generation and load of eliminated nodes:

1. Generation and load assumed constant current.
2. Generation assumed constant current, load assumed constant admittance.
3. Generation and load assumed constant admittance

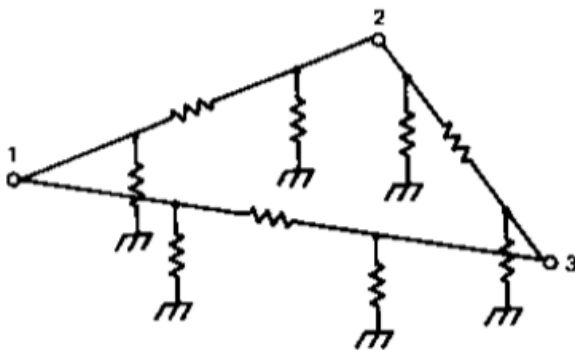
These options are illustrated by the examples in the figure below. For simplicity, losses are ignored and a DC model is used.

In all examples, the equivalent branch flow represents the sum of three components:

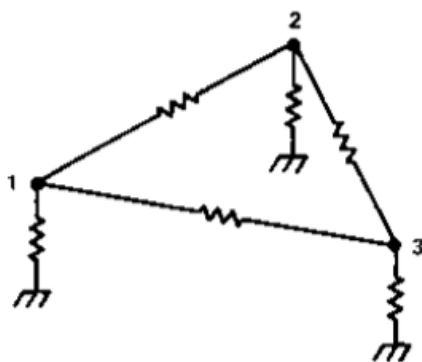
1. The power looping into the eliminated system and back out to the retained system.



a) Original four-node network



b) Equivalent reduced network  
(node 0 eliminated)



c) Actual representation  
or equivalent pi-branches:  
legs-to-ground consolidat-  
ed with nodal suscep-  
tance.

Fig. 2.15.1: Equivalent Pi-Branches

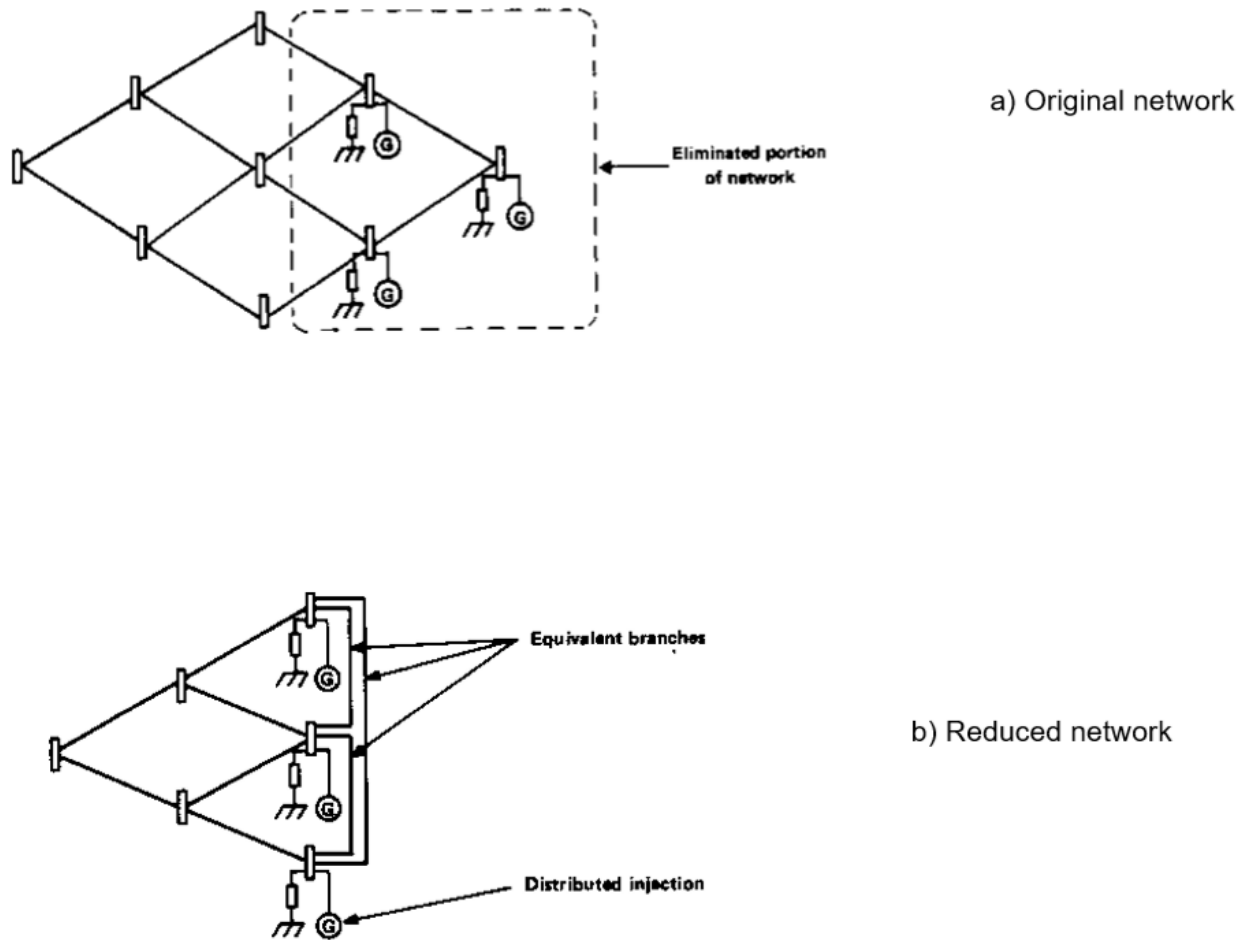
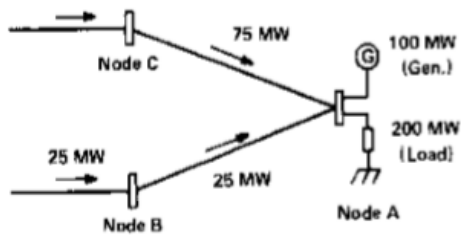
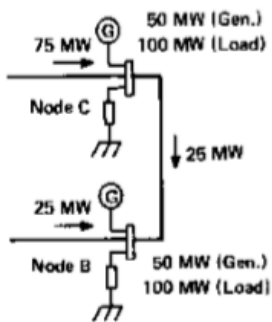


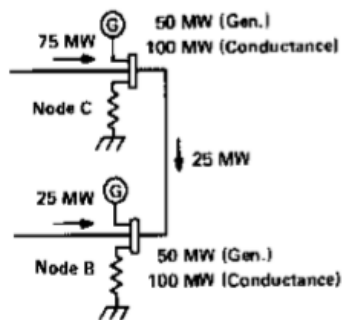
Fig. 2.15.2: Network Reduction



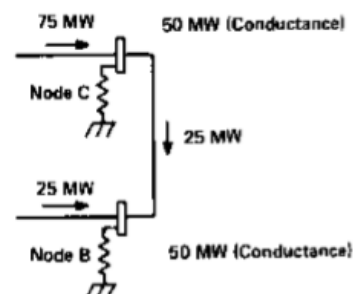
a) Original network



b) Network with node A eliminated with option 1 assuming equal branch admittance



c) Network with node A eliminated with option 2 (loads replaced with distributed bus susceptance)



d) Network with node A eliminated with option 3 (net load and generation replaced with distributed bus susceptance)

Fig. 2.15.3: Eliminated Node Generation and Load Disposal

2. The flow of generation within the retained system through an envelope node to the distributed loads of the equivalent system.
3. The flow of distributed generation on the envelope nodes to loads within the retained system.

The difference between network reduction and network cutting is seen in component 1 above. This component is found by assuming all generations and loads within the eliminated system are identically zero. Thus, in equation Eq.2.15.5,  $I_1 = 0$  and

$$[I_2]_{eq} = [I_2] \quad (2.15.7)$$

The branch flow in this case is strictly due to the differences of voltages between envelope nodes, i. e., looping. This is the important distinction between reduction and cutting.

## 2.15.2 Description of Reduction

The nodal admittance matrix is retrieved from the base file and selective elimination is performed first upon all eliminated nodes and then partially upon the retained nodes.

A complete pass merges the data from the reduced current matrix and the reduced admittance matrix with the system data from the base file. Eliminated data is purged and equivalent branches and distributed injections are added to the data. Concurrent with this pass, a simple injection check is performed on the envelope nodes. The net injection of the nodes is compared with the original values. Any mismatches are errors and will be flagged.

Special consideration is given to the distributed shunt admittances. The envelope nodes which receive these admittances may be subtype Q. In such a case, this portion of the shunt admittance is not adjustable for voltage control. To accomplish this, a special continuation bus subtype, +A, was created, to which all distributed injections and shunt admittances are added. In the power flow output listings, this portion of the admittance is flagged “EQUIVALENT” to confer distinction over other adjustable susceptances.

Upon completion of the reduction, all equivalent branches and injections are added to the system data and all data in the eliminated system is permanently deleted. A new base case is created, updated, and must be solved. Thus, a /SOLUTION and /NEW\_BASE command must follow any network reduction. If desired, subsequent changes could be applied.

An example setup for doing a network reduction with bpf is shown in [Reduction Case Example](#).

The program is written such that the method of determining the base case is irrelevant to the options that may be performed. Once a base case is defined and solved, any of the options for a base case may be exercised. Thus, further network reduction could be performed upon the already reduced system. Proper position of the appropriate control cards give the user complete flexibility.

## 2.15.3 Program Control Options

There are seven options that the user may exercise; they are all specified by qualifiers within the /REDUCTION command set.

1. Admittance cutoff (MIN\_EQUIV\_Y).
2. Disposition of injections (elimination mode: ELIM\_MODE, final mode: ULT\_MODE).
3. Retain generators by “REI” scheme (REI\_CLUSTERS, ELIM\_MODE).
4. Retain area interchange nodes (KEEP\_AI\_SYSTEM).
5. Retain all generators (RETAIN\_GEN).
6. Optimal network determination (OPTIMAL\_REDU)
7. Minimum generator cutoff for “REI” equivalent (ELIM\_MODE).

These options are discussed in the following paragraphs.

### Admittance Cutoff

Assume that the retained network has  $n$  nodes in which  $m$  ( $m < n$ ) nodes define the envelope. Assume furthermore that there is a path from any envelope node into the eliminated network and back to any other envelope node. Then, inherent in the reduction, there will result an equivalent branch between any pair of envelope nodes. The total equivalent branches added will be

$$\frac{m(m-1)}{2}$$

This total can become large. Many equivalent branches added between the most relatively remote pair of envelope nodes will have a branch impedance excessively large and presumed negligible. The admittance cutoff is a parameter that the user may choose in eliminating these branches from the generated reduced system data. It is defined as

$$\text{Admittance Cutoff} = \frac{1}{R + jX_{\text{equivalent}}}$$

For example, the value 0.001 will exclude all equivalent branches with an equivalent impedance of 1000 p.u. ohms or more.

### Disposition of Injections

The disposition of injections of eliminated nodes has, in all, 12 different combinations of options. Each combination is unique with its inherent advantages and disadvantages.

The injections are divided into three parts: (1) generation, (2) load, and (3) shunt admittance. Each of these injections has two modes of disposition during reduction and three additional modes ultimately. The transition of a single injection is shown in below.

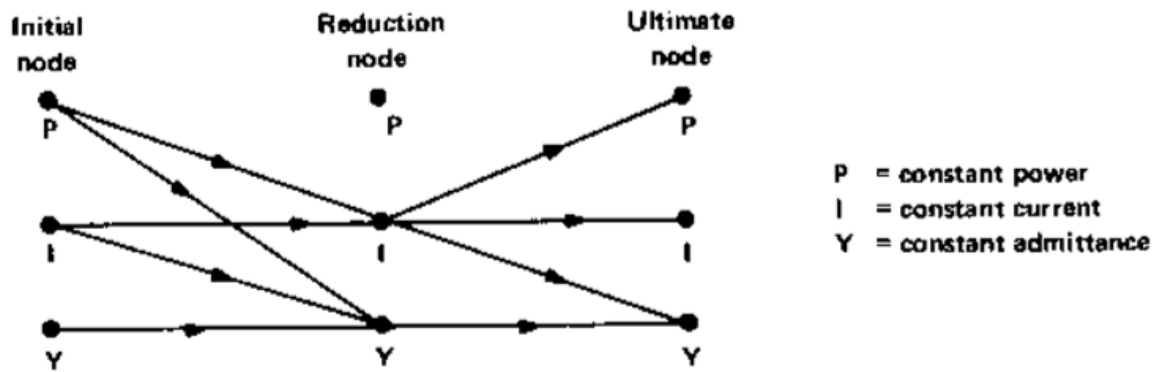


Fig. 2.15.4: Injection Disposition of Eliminated Nodes



## Reduction Mode

The two possible modes of disposition during reduction are constant current and constant admittance. A summary of each option with its inherent characteristics follows:

Generation:

- *Constant current* has improved convergence characteristics; retains identity as generation.
- *Constant admittance* appears as fictitious negative impedance; has major effect on passive equivalent network.

Load:

- *Constant current* retains identity of load
- *Constant admittance* has slightly better convergence characteristics; has major effect on passive equivalent network.

Shunt admittance:

- *Constant current* avoids negative impedances in the equivalent network.
- *Constant admittance* preserves the identity of the passive network.

## Ultimate Mode

After equation Eq.2.15.6 has been solved for  $I_{2eq}$  and  $Y_{22eq}$ , the question arises: How should the separate components of  $I_{2eq}$  be disposed to their ultimate state?

From Figure 5-4 we have three options to dispose  $I$ ; namely  $P$ ,  $I$ , or  $Y$ . A summary of each is described below.

Generation and load:

- *Constant power* is typical and preserves the identity of generation or load.
- *Constant current* is preferred and has superior convergence characteristics.
- *Constant admittance* is the least preferred and has poor accuracy.

Shunt admittance:

- *Constant power* is atypical and is available for compatibility only.
- *Constant current* is the same as constant power.
- *Constant admittance* is preferred and is most realistic to a passive network.

## Summary

The recommended disposition is therefore summarized in the table below.

Table 2.15.1: Disposition of Components

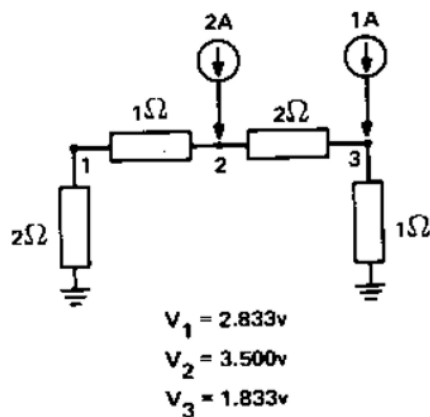
Quantity	Reduction Mode	Ultimate Mode
Generation	I	I
Load	I	I
Shunt Admittance	Y	Y

### 2.15.4 REI Equivalent

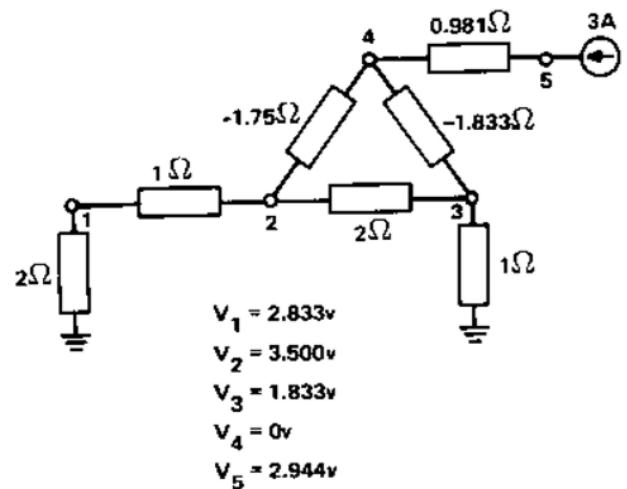
One disadvantage inherent in reduction involves the properties of eliminated generators. The power injections are converted into current injections using the steady-state solution voltages. Every eliminated generator therefore becomes identical to a bus with fixed real and reactive injection, but without fixed voltages typical of BQ nodes with reactive inequality constraints. Consequently, the eliminated system loses its voltage regulation capacity. System changes near the border nodes of reduced equivalent systems often converge to voltage profiles quite different from the full system. The obvious remedy is to reinstate these generators or, alternately, to preserve their regulating characteristics.

The normal option of retaining generators has been unsatisfactory. A network having 1600 nodes that is reduced to a system of 600 nodes may also include about 200 retained generators and an extremely disproportionate number of equivalent branches. The reduced equivalent system is typically about 75 percent of the size of the full network and nearly defeats any merits gained in reduction.

The “REI” equivalent is an innovative alternate to preserving eliminated generators directly. The initials mean Radial Equivalent Independent. It is a simple scheme in which several eliminated generators are connected to a common ground node having zero voltage but isolated from the ground of the rest of the system. This ground node is then tied directly to an equivalent generator. The branch admittance from the ground node to all the generators and to the equivalent node are determined such that no real or reactive power is gained or lost. A simple resistive network as shown in the figure below demonstrates the procedure.



a) Network having current sources at nodes 2 and 3



b) REI network in which the equivalent currents are coalesced into a single node

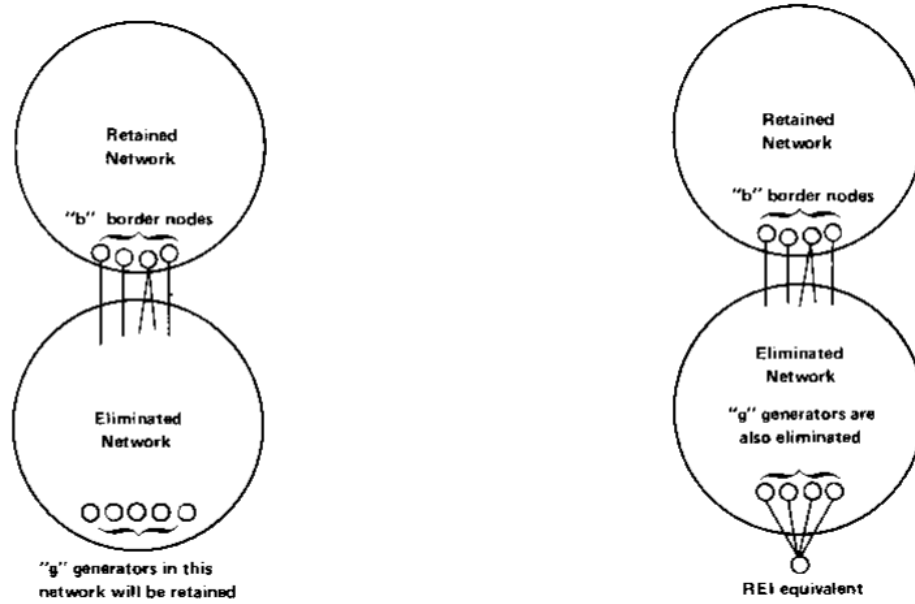
Fig. 2.15.5: Example of Network and Its REI Equivalent

The example above has moved the current injections at nodes 2 and 3 in (a) back to an equivalent node 5, which has 3A injection. The current flowing from 4 to 2 is 2A; from 4 to 3 is 1A. The power loss in the branches is

$$\begin{aligned}
 P &= I_{42}^2 R_{42} + I_{43}^2 R_{43} + I_{54}^2 R_{54} \\
 &= 2^2(-1.75) + 1^2(-1.833) + 3^2(0.981) = 0 \text{ watts}
 \end{aligned}$$

This zero power loss is deliberate. It is termed the zero power balance. Introduction of the REI system has not changed the total system losses.

Although this example demonstrates the procedure involved, it does not illustrate the merits in preserving sparsity. Only in large networks does this become apparent. Consider the network in the figure below.



(a) Direct

Network schematic in which the eliminated network contains "g" generator nodes. These generators will be excluded from eliminated network and therefore be appended into the retained network.

(b) Indirect

Same network as (a) except the "g" generator nodes will be tied to one REI equivalent. The equivalent node will be appended to the retained network and the "g" generators will be eliminated.

Fig. 2.15.6: Direct and Indirect Generator Preservation in Eliminated Network

The letter "b" in the figure above is the number of border nodes in the network. The letter "g" is the number of generators in the eliminated network. If they are saved directly and the rest of the network is eliminated, the number of equivalent branches is at worst.

$$N_{br} = \frac{g(g-1)}{2} + gb + \frac{b(b-1)}{2}$$

However, if an REI equivalent is obtained instead,  $g = 1$  and

$$N'_{br} = b + \frac{b(b-1)}{2}$$

Typical numerical values are  $g = 10$ ,  $b = 15$ . Then  $N_{br} = 225$  and  $N'_{br} = 120$ . This is nearly a 50 percent reduction in the number of equivalent branches.

### 2.15.5 REI Clusters

In the figure of the previous example, all of the border nodes will be normally fully interconnected. The total number of branches is easily computed by  $b(b-1)/2$  where  $b$  is the number of border nodes. If  $b$  is large, say 50, then  $50*49/2 = 1225$  equivalent branches will result during this elimination. (The same number would occur without the REI equivalents.) However, if the area interchange system is retained, the eliminated system becomes clustered with a dramatic reduction in the number of equivalent branches.

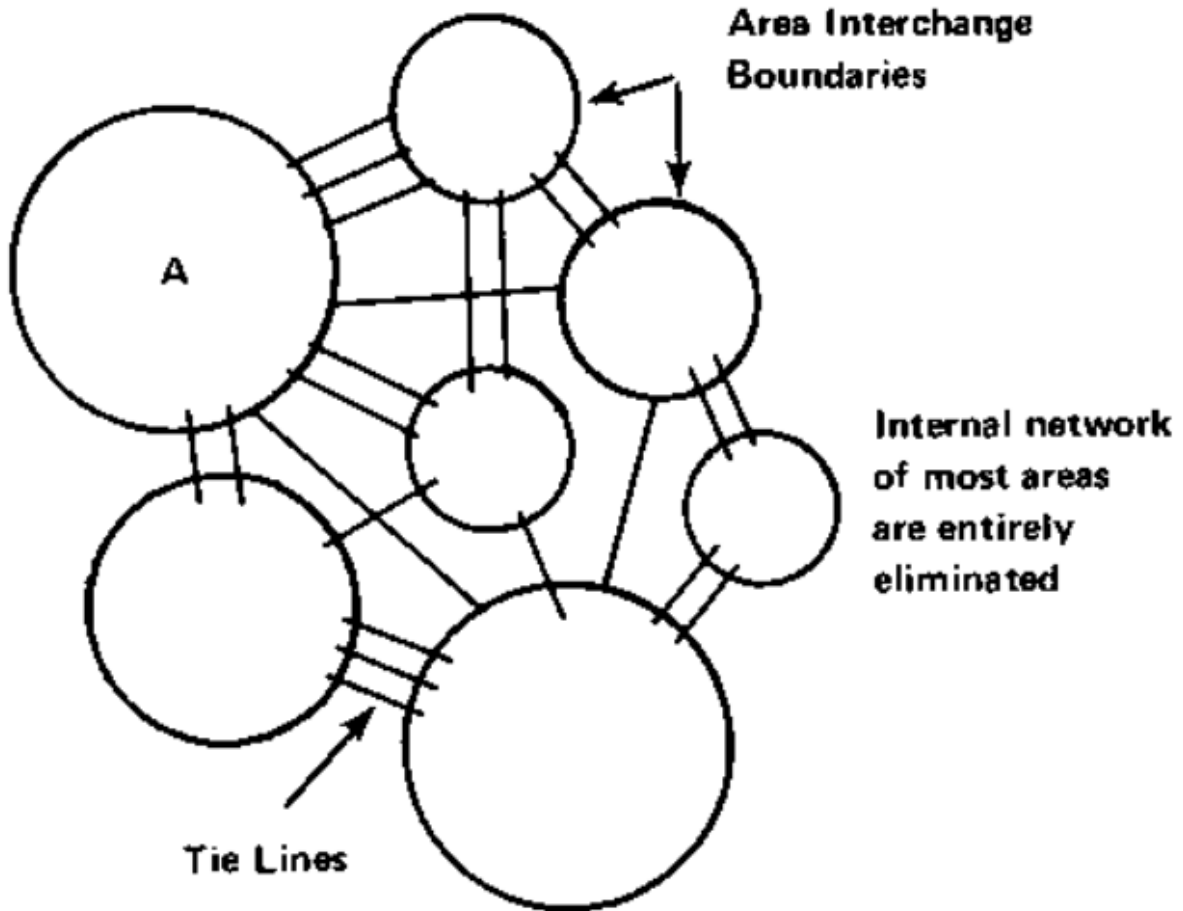


Fig. 2.15.7: Small System Network (Areas and Tie Lines Shown)

A typical equivalent of the figure above might normally retain area A in full detail, and would include all the tie line nodes and area slack buses of the remaining system. The rest of the system is replaced with an equivalent. This area interchange system permits the eliminated system to be assigned clusters (a minimum of one cluster per area). Each cluster is assigned an “REI” equivalent generator to replace all the eliminated generators.

Defining clusters does not decrease the number of eliminated generators nor decrease the number of border nodes. However, it isolates the interconnections from one REI equivalent to another. For example, suppose that the previous example was split into two clusters defined with the following bus counts:

$$\begin{aligned}
 &= 5g_2 = 5g_1 + g_2 = 1 \\
 &= 7b_2 = 8b_1 + b_2 = 1 \\
 N'_{br} &= \left(b_1 + \frac{b_1(b_1 - 1)}{2}\right) + b_2 + \left(\frac{b_2(b_2 - 1)}{2}\right) = 64
 \end{aligned}$$



### 2.15.7 Using REI Equivalents

The ELIM\_MODE command requests REI equivalencing of eliminated generators. An additional parameter specifies the minimum PGEN for a bus to be normally coalesced. Larger values of PGEN will exclude the smaller generators from being coalesced. Care should be exercised when requesting coalesced generators, since table overflow will occur if more than 99 generators are coalesced into a single equivalent.

The equivalent generators are made type BQ with a scheduled voltage computed for zero power balance. Names such as “EQUIV 1” are program generated and pertain to the clusters in which they reside. However, if “COH” data is submitted, the generator names are as specified in the data. The zone of an equivalent generator is randomly selected to be any zone of one of the coalesced generator.

### 2.15.8 Optimal Network Determination

When not using the REI equivalent, the number of equivalent branches may become excessively large and compromise any advantages obtained by network reduction, unless the retained network is judiciously chosen. If the optimal feature is selected using the OPTIMAL\_REDU command, the retained network specified by the input data will be interpreted as the “minimum” retained system. Nodes selected for actual elimination will contribute to a definite reduction in system size. Thus, the optimal feature will expand the retained system if necessary to minimize the network size.

### 2.15.9 REDUCTION Command

This control card requests a network reduction to be performed on the base case or change case that has been defined. It generates an unsolved, reduced equivalent base case. Other commands must eventually follow to obtain the new solution and any desired outputs. However, system data could follow the /REDUCTION to update the reduced case.

The reduced system is composed of internal and enveloping retained nodes. The internal nodes undergo no change during the reduction. The enveloping nodes are those which have branches into the reduced system. However, these branches are replaced with equivalent branches connected to other enveloping nodes which simulate the reduced system. In addition, any nodal injection is distributed to the enveloping nodes in the form of continuation bus data.

All distributed injections and equivalent branches are flagged with the ownership code \*\*\*. This emphasizes their fictitious nature; however, they are valid system data and should be treated as such.

## 2.16 Retained Network Definition

The network to be reduced is obtained from a solved base case. The retained system is defined by the user in the INPUT file. Five options are available to the user as shown in the figure below.

Each of the options 1 through 5 are affected by the unique control cards described in upper case characters in the blocks in the figure above. Their formats are described in the following paragraphs.

### 2.16.1 Reduction Cards

The reduction cards specify the retained systems. There are several options available. 1. Zone Selection. The zones which normally define geographical areas may be used to define the retained system. All buses within these zones are then retained. The format is essentially “free-field”. Commas separate the different zones and a period terminates the list. Any blanks are ignored unless immediately preceded by a comma or a period. The command >SAVE\_ZONES must begin in column 1. If the zone list is too long to be contained on a single card, continuation cards must be used, in which a blank appears in column 1 and the zone list continued. 2. Base kV Selection. The retained system may be selected as the set of all buses with base kV’s matching the specified set. As with the previous card, the format is essentially free-field. However, decimal points may be part of the base kV, necessitating a double period to terminate

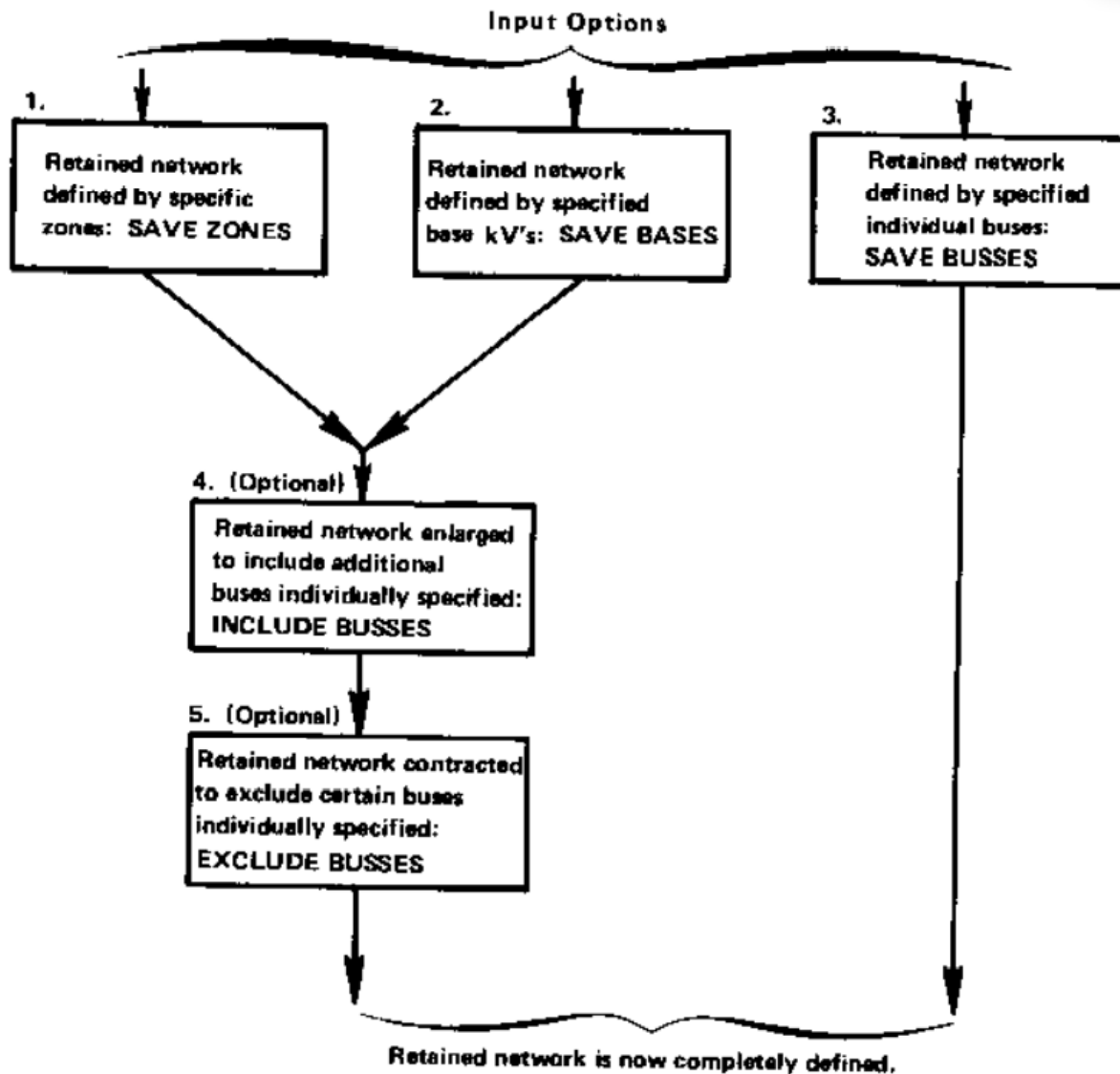


Fig. 2.16.1: Retained Network Input Options

the list. The command >SAVE\_BASES must begin in column 1. Continuation cards are permitted. If they are used, a blank must appear in column 1 as the base kV list continues. 3. Zone Selection Subjected to Base kV's. This is a combination of steps 1 and 2 described previously. Each of the zones selected is further stipulated to certain base kV's. Recognition is acknowledged that different zone/base kV combinations may be desirable, and several zone-base KV cards are permitted. The format is similar to steps 1 and 2 above. The difference is that the last zone (there may be only one) is followed by a comma and the words >SAVE\_BASES. Continuation cards are permitted. 4. Inclusion and Exclusion of Buses to Retained System. The general criteria of saved zones, saved bases, or both is recognized as too general to adequately define all retained systems. Provisions permit the system thus defined to be enlarged or contracted by specifying individual buses which are to be included into or excluded from the retained system. The retained system is expanded with the control card. >INCLUDE\_BUSES beginning in column 1 and followed with separate bus cards. The bus card need only contain the B in column 1, and the bus name and base kV in columns 7-18. All other fields are ignored. The list of included buses is terminated when a card without a B in column1 is encountered. The retained system is contracted with the control card >EXCLUDE\_BUSES beginning in column 1 and followed with separate bus cards as described previously. 5. Individual Bus Selection. In lieu of options 1. through 4.described previously, the retained system could be identified by individually specified bus cards. This request is initiated with the control card >SAVE\_BUSES beginning in column 1. Each retained bus is then individually specified with a bus card. Only the bus identification fields need be specified, that is, the B in column 1 and the bus name and base kV in columns 7-18. All other fields will be ignored. The list continues until a card without a B in column1 is encountered.

## 2.16.2 Input listings

Following the program control card and subsequent reduction cards, the retained system is defined and two listings are produced. The first listing defines the retained network. Each bus name will be preceded with one or two of the following characters: Blank Retained bus is internal and unchanged. E Retained bus is an envelope node and will receive distributed injections and equivalent branches. X Retained bus is also an envelope node. However, everyone of the adjacent nodes is eliminated. This is acceptable but may not be intentional and is intended as an informative diagnostic. \* Retained bus is selected by the optimal feature. P Retained bus is a generator. The second listing defines all eliminated nodes.

## 2.17 Voltage Limits and Starting Voltages

The following table allows you to find global and starting voltages for a bus given the base kV and zone. These values are hard-coded in the program and cannot be altered by the user.



Table 2.17.1: Global Voltage Limits and Starting Voltages

Base kV Range		Zone Restrictions	Global V Limits		Starting V	
From	To		V_min (p.u.)	V_max (p.u.)	Gen Buses (40%)	Load Buses (20%)
0.1	6.5		0.950	1.052	1.011	1.032
6.6	6.6	M1	0.950	1.065	1.019	1.042
6.6	49.9		0.950	1.052	1.011	1.032
50.0	50.0	16	1.100	1.200	1.160	1.180
50.0	59.9		0.950	1.052	1.011	1.032
60.0	60.0	17	0.950	1.100	1.040	1.070
60.0	62.9		0.950	1.052	1.011	1.032
63.0	63.0	20	0.930	1.080	1.020	1.050
63.0	99.9		0.950	1.052	1.011	1.032
100.0	100.0	16 M5	1.100	1.200	1.160	1.180
100.0	100.0	M4	0.950	1.070	1.022	1.046
100.0	114.9		0.950	1.052	1.011	1.032
115.0	115.0	M4	0.950	1.070	1.022	1.046
115.0	131.9		0.950	1.052	1.011	1.032
132.0	161.0	17 20	0.950	1.090	1.034	1.062
132.0	199.9		0.950	1.052	1.011	1.032
200.0	200.0	16	1.100	1.200	1.160	1.180
200.0	229.9		0.950	1.052	1.011	1.032
230.0	230.0	17 20	0.950	1.070	1.022	1.046
230.0	499.9		0.950	1.052	1.011	1.032
500.0	500.0		1.000	1.100	1.060	1.080
500.1	1099.9	1	0.950	1.052	1.011	1.032
1100.0	1100.0	1	1.000	1.100	1.060	1.080
1100.1	9999.9	1	0.950	1.052	1.011	1.032

### 2.17.1 Notes

1. Twenty percent (20%) and forty percent (40%) starting voltages are the percentages from  $V_{max}$  to  $V_{min}$ :

$$V_{start} = V_{max} * (1 - \%pct) + V_{min} * \%pct$$

“0%” starts at  $V_{max}$ ; “100%” starts at  $V_{min}$ .

Ideal starting percentages would be a value which lies closest to the final voltages. For peak load cases, a generator percentage of 30% is better; for light loads, 50% is better. The current 40% is a trade-off.

2. Zone restrictions mean that base kV's are also subject to zones. If the zone restriction list has non-blank entities, only base kV's within the range and zone list zones apply. Conversely, if the zone restriction list is blank, then there are no zone restrictions: all base kV's within the range apply.
3. To find global and starting voltages for a bus with Base XXX and Zone ZZ, proceed as follows:
  - a. Find the encompassing base kV:  
 $\text{From\_base} \leq \text{XXX} \leq \text{To\_base}$   
 There will be one or more such ranges. Zone restrictions may qualify the base range.

- b. If zone restrictions apply, check whether zone ZZ qualifies. If true, then the pertinent entity has been located.

If false, proceed to the next encompassing base kV and repeat step b. All encompassing base ranges terminate with an inclusive blank zone list, which ultimately defaults to “no restrictions”.

## 2.18 Developer Notes

This section contains notes for developers working with, enhancing, or debugging the IPF codebase.

### 2.18.1 Debugging

You can list names of functions or subroutines to debug in a DBGIN file in .fil passed to tsp. Debug output is printed to file defined by DBGOUT. For example, the following routines can be listed: CNTRL, CNTRLA, FFREAD, INITL1, INITL4, INPUT1, INT3FA, LSREAD, MATMOD, NAMBAS, NOUT1, NOUT2, RVREAD, SOLN, SWINGM, TAPEWK, WRTHIS. These Fortran subroutines will have their debug enabled.

### 2.18.2 bpf Flowchart

Below is a high level flowchart of the batch power flow process.

### 2.18.3 tsp Flowchart

Below is a high level flowchart of the transient stability program.

### 2.18.4 Variables

Reading IPF code can be challenging. Fortran 77 was the programming language of the 70s and 80s. Today there are small percentage of developers that know Fortran and even fewer that have experience writing significant amounts of it. Though it is important to note that the language is still used in several well known projects such as the [Linear Algebra PACKage \(LAPACK\)](#).

Fortran developers must use very concise variable and function/subroutine names to make sure lines fit in the 80 character limit and original limitations on 8 character file names. Additionally, IPF code base makes heavy use of common blocks to pass data between functions (these are basically like blocks of data since Fortran has no concept of an object or even a struct as in C) as well as goto statements. This can make any given piece of code overwhelming to a newcomer. For example, you might encounter something like this and feel completely lost:

```
kbrknt = mptr / 2
next1 = mptr + 1
next2 = mptr + 1
last1 = nsize
last2 = nsize
do k = next1, last1
    kolum1(k) = 0
    kordr1(k) = k+1
enddo
kordr1(last1) = 0
```

(continues on next page)

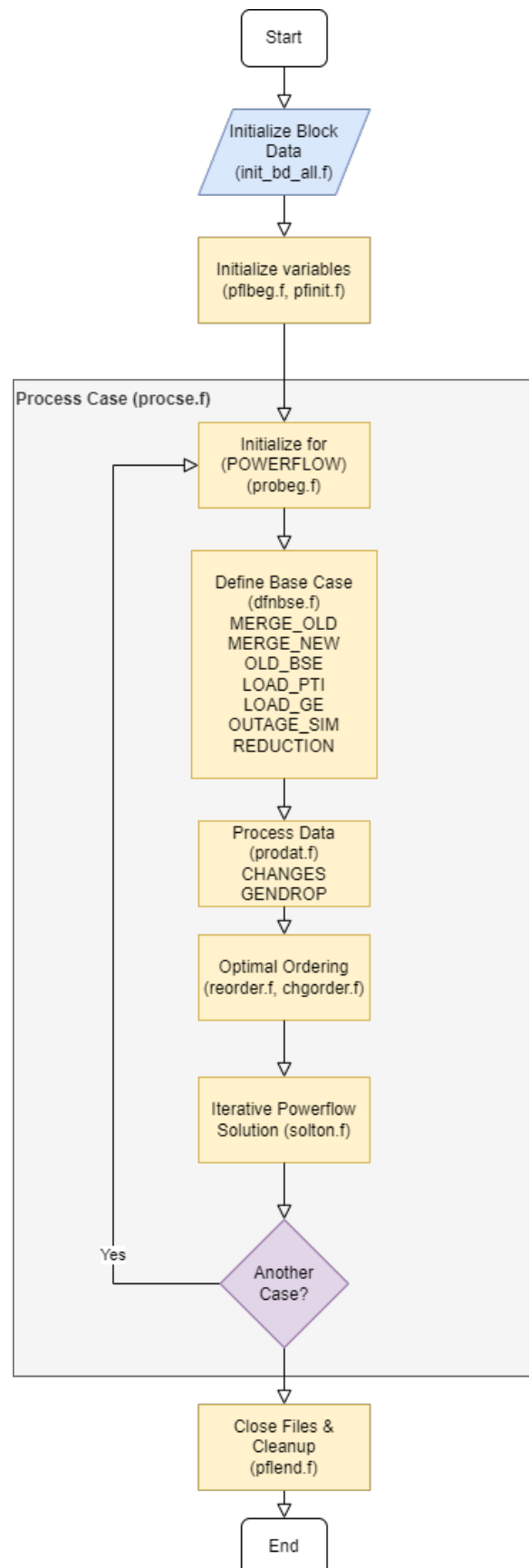


Fig. 2.18.1: IPF bpf Flowchart

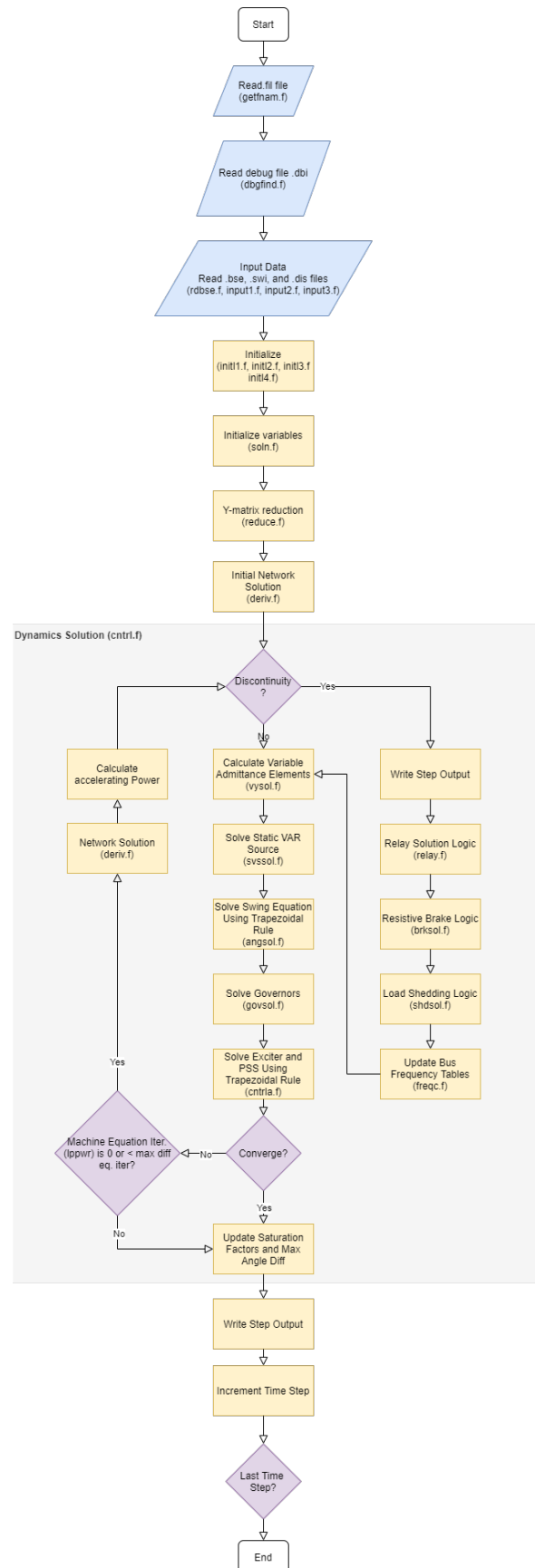


Fig. 2.18.2: IPF bpf Flowchart

(continued from previous page)

```

do k = 1, ntot
  jorder(k) = kownt(2,k)
  loc2(k) = loc1(k)
enddo
do k = 1, nsize
  kordr2(k) = kordr1(k)
  kolum2(k) = kolum1(k)
enddo

```

The table below contains a list of more descriptive names for variables. This can be used as a reference when getting started.

Variable	Description
datot	Delta area real power total. Total area real power mismatch for power flow
dptot	Delta real power total. Total real power mismatch for power flow iteration
dqtot	Delta reactive power total. Total reactive power mismatch for power flow iteration
dtot	Delta transformer MVAR. Total transformer apparent power mismatch for power flow iteration
icount	Count of solution truncation adjustments
ikecl	Jacobian matrix storage size
ittot	Power flow iteration counter
kownt	Unsolved buses count
kowntb	Unsolved auto transformers
kowntc	Unsolved areas
lppwr	Iteration for solving machine equations
eyr	Bus per unit voltage real component. "A" in "A + jB"
eyi	Bus per unit voltage imaginary component. "B" in "A + jB"
isg	Number of synchronous generators
idsw	Discontinuity state

## 2.18.5 Command Subroutines

This section has a information about which Fortran subroutines to take a look at if you're trying to debug a certain PFC or PCL command.

```

/INITIALIZE

  processed by  p_pfini_

/NETWORK_DATA, FILE = <filename>

  processed by  p_gtnetdat_

/OLD_BASE, FILE = <* | filename> [, CASE = <casename>]
                    [, REBUILD = < on | off> ]

  processed by  cmd_parse.c
                p_gtbase.f
                ctlpow.f

/CHANGES, FILE = <* | filename>

```

(continues on next page)

(continued from previous page)

```

        processed by cmd_parse.c
                   p_change.f
                   ctlpow.f

/SOLUTION
> BASE_SOLUTION
> DEBUG, BUS = ON,(page 3-51 to 3-56)

        processed by cmd_parse.c
                   p_solton.f
                   ctlpow.f

/GET_DATA, TYPE = INPUT
A    <areaname>  to be added 22 July by wlp
I    <area1 area2> to be added 22 July by wlp
B    <busname, etc> returns all data associatated with bus
+    <busname, etc> returns all data if id fields have wild cards
      (type - column 2, owner, columns 3-5, and code-year
      columns 20-21)
X    <busname, etc>
L    <bus1 bus2, etc> returns all paralles if id is wild card (*)
      returns all sections if section is 0
T    <bus1 bus2, etc>
R    <bus1 bus2, etc>
E    <bus1 bus2, etc>

        processed by cmd_parse.c
                   p_gtdata.f
                   gtinput.f

/GET_DATA, TYPE = BUS_VOLTAGES

        processed by cmd_parse.c
                   p_gtdata.f
                   bus_voltages.f

/GET_DATA, TYPE = BUS_LIST [ FROM BUS_DATA ]
      WHERE AREAS = <area1>, <area2>, etc AND
            ZONES = <zone1>, <zone2>, etc AND
            OWNERS = <own1>, <own2>, etc AND
            BASEKV = base1
                  < base ( example < 115.0 means all base
                        kv's less than or equal to 115.0)
                  > base ( example > 115.0 means all base
                        kv's greater than or equal to 115.0)
            base1 < base2 (all bases between base1 and
                        base 2)
            base2 > base1 (same as above)
      TYPE = B , BE, BS, BC, BD ,BV ,BQ ,BG ,BO ,BT ,BX ,
            BM ,L ,LD ,LM , ,E ,T ,TP ,R
      BUS = "<busname>" (quotes are necesary)
      AFTER_BUS = "<busname>" ( to do 22 July by wlp)

```

(continues on next page)

(continued from previous page)

```

                ALL
                LOADING = (<min> <max>)

        processed by cmd_parse.c
                    p_gtdata.f
                    bus_list.f
                    gtfltr.f

/GET_DATA, TYPE = A_DATA

        processed by cmd_parse.c
                    p_gtdata.f
                    a_data.f

/GET_DATA, TYPE = I_DATA

        processed by cmd_parse.c
                    p_gtdata.f
                    i_data.f

/GET_DATA, TYPE = BSEKV_LIST

        processed by cmd_parse.c
                    p_gtdata.f
                    bsekvlst.f

/GET_DATA, TYPE = AREA_LIST

        processed by cmd_parse.c
                    p_gtdata.f
                    area_list.f

/GET_DATA, TYPE = OWNER_LIST

        processed by cmd_parse.c
                    p_gtdata.f
                    owner_list.f

/GET_DATA, TYPE = BUS_EXISTS, BUS = "bus_name"

        processed by cmd_parse.c
                    p_gtdata.f
                    ex_bus.f
        return status: status = 0 : bus does not exist
                        1 : bus exists

/GET_DATA, TYPE = FILE_EXISTS, FILE = <file_name>

        processed by cmd_parse.c
                    p_gtdata.f
                    ex_file.f
        return status: status = 0 : file does not exist

```

(continues on next page)

(continued from previous page)

```

                                1 : file exists
/GET_DATA, TYPE = CONNECTION
B    <busname, etc> returns all connection data associatated with bus

    processed by cmd_parse.c
                p_gtdata.f
                ex_file.f
    return status: status = 0 : file does not exist
                    1 : file exists

/GET_DATA, TYPE = ZONE_LIST

    processed by cmd_parse.c
                p_gtdata.f
                zone_list.f

/GET_DATA, TYPE = OUTAGES

    processed by cmd_parse.c
                p_gtdata.f
                gtoutage.f

/GET_DATA, TYPE = LINE_IMPEDANCE_CALCULATION
    UNITS = < ENGLISH | METRIC >,
    DISTANCE = < miles | km >
    BASEKV = <basekv>,
    BASEMVA = <basemva>,
    FREQUENCY = <freq>

    processed by cmd_parse.c
                p_gtdata.f
                p_lic.f
                linimp.f

/GET_DATA, TYPE = output

    processed by cmd_parse.c
                p_gtdata.f
                gtoutput.f

/GET_DATA, TYPE = INITIALIZE_DEF

    processed by cmd_parse.c
                p_gtdata.f
                p_initdef.f

/GET_DATA, TYPE = LOAD_DEFINE
> DEFINE ...

    processed by cmd_parse.c
                p_gtdata.f
                p_loaddef.f

```

(continues on next page)



(continued from previous page)

```

/GET_DATA, TYPE = SUB_DEFINE, SOURCE = BASE
                                ALTERNATE_BASE

```

```

    processed by cmd_parse.c
                p_gtdata.f
                p_subdef.f

```

```

/GET_DATA, TYPE = LOAD_AREA

```

```

    processed by cmd_parse.c
                p_gtdata.f
                p_ldardata.f
    return status: status = 0 : success
                      1 : errors

```

```

/GET_DATA, TYPE = AREA_DATA

```

```

A <areaname>

```

```

    processed by cmd_parse.c
                p_gtdata.f
                gtardata
    return status: status = 0 : success
                      1 : errors

```

```

/REPORTS, SELECT BUS_BR_INPUT [ FROM BUS_DATA ]
    [ OUTPUT = <filename> ]
    WHERE (repeat filter from BUS_LIST)

```

```

    processed by cmd_parse.c
                p_report.f
                busbrinrpt.f
                gtfltr.f

```

```

/REPORTS, SELECT BUS_BR_OUTPUT [ FROM BUS_DATA ]
    [ OUTPUT = <filename> ]
    WHERE (repeat filter from BUS_LIST)

```

```

    processed by cmd_parse.c
                p_report.f
                busbrotrpt.f
                gtfltr.f

```

```

/REPORTS, SELECT OVERLOADED_LINES [ FROM BUS_DATA ]
    [ OUTPUT = <filename> ]
    WHERE (repeat filter from BUS_LIST)

```

```

    processed by cmd_parse.c
                p_report.f
                ovldlnsrpt.f
                gtfltr.f

```

(continues on next page)

(continued from previous page)

```
/REPORTS, SELECT OVERLOADED_TXS [ FROM BUS_DATA ]
      [ OUTPUT = <filename> ]
      WHERE (repeat filter from BUS_LIST)

      processed by cmd_parse.c
                  p_report.f
                  ovldtxsrpt.f
                  gtfltr.f

/REPORTS, SELECT BUS_UVOV [ FROM BUS_DATA ]
      [ OUTPUT = <filename> ]
      WHERE (repeat filter from BUS_LIST)

      processed by cmd_parse.c
                  p_report.f
                  busuvovrpt.f
                  gtfltr.f

/REPORTS, SELECT
      (GUI-customized reports, but not completed!)
      processed by cmd_parse.c
                  p_report.f
                  outputrpt.f
                  gtfltr.f

/NEW_BASE, FILE = <filename>, CASE = <casename>
      processed by cmd_parse.c
                  p_newbse.f
                  ctlpow.f

/QUIT

      processed by p_pfexit_

/EXIT

      processed by p_pfexit_

/CFLOW_GUI

      processed by p_cflow_gui

/INITDEF

      processed by p_initdef_

/LOADDEF

      processed by p_loaddef_

/SUBDEF
```

(continues on next page)

(continued from previous page)

processed by p\_subdef\_

## 2.18.6 WSCC Specifications

These functional specifications are here for more historical reasons than present day usefulness but they are from Western Systems Coordinating Council Load Flow and Stability Computer Programs department. They were used during initial development of IPS.

### General, Load Flow, & Stability

- A. WSCC will consider any deviations and alternates of these specifications, provided the details of such deviations and alternates are explained by the supplier.
- B. Load Flow and Stability programs shall be compatible. Load flow shall accept card input or tape input from previous cases. Stability program shall accept load flow output and system data directly from load flow history tape of any case. Load flow and stability shall also run "back-to-back" if desired, that is, stability program run as a continuation of load flow without operator intervention to initiate stability run.
- C. Program shall be written in a standard Fortran language for ease of modification except for relocatable Fortran subroutines.
- D. Programs shall be completely documented including full explanation of theory and equations upon which calculations are based as well as the Fortran listings. Complete users manual and program documentation shall be supplied with delivery of programs. A glossary of terms shall be included in the documentation.
- E. Programs shall be written for computer acceptable to the Council.
- F. Reduced computer storage capacity shall result in reduced program limits rather than elimination of program features.
- G. Program features shall not be at the expense of excessive input manipulation or machine time.
- H. Diagnostics shall be provided in program operation to reveal cause of program stoppage or failure to converge. Included in diagnostic shall be maximum real and quadrature components of power and voltage change and names of buses on which they occur. Bus mismatch shall be available as diagnostic at user's option. Diagnostics shall be available at each iteration at user's option.

### Load Flow Program

- A. System Limits
  - 1. 2000 buses
  - 2. 3000 branches, 1000 of which may be transformers with fixed or LTC taps.
  - 3. 50 phase shifters
  - 4. 60 interchange control areas.
  - 5. Voltage at 400 buses controlled remotely by generation or transformer LTC.
  - 6. 90 loss accumulation areas other than interchange areas.
  - 7. 10 two-terminal and 5 three-terminal DC lines.
- B. Buses

1. Machines, loads, shunt capacitors, and shunt reactors shall all be representable separately at each bus.
2. Loads shall be representable as scheduled mw and mvar.
3. Shunt admittances shall be representable in per-unit as follows:
  - (a) Fixed admittance
  - (b) Switched capacitor in steps with on and off voltages
  - (c) Switched reactor in steps with on and off voltages
4. Machines shall be representable as follows:
  - (a) Scheduled voltage magnitude and angle (swing).
  - (b) Scheduled mw and voltage magnitude, with or without mvar limits. In event mvar limit is reached, voltage schedule is no longer held.
  - (c) Scheduled mw and mvar, with or without voltage limits. In event voltage limit is reached, mvar schedule is no longer held.
  - (d) Scheduled mw and variable mvar to hold voltage on remote bus. Provision shall be made at 50 voltage controlled buses for the controlling action to be shared in scheduled proportion among the reactive outputs of up to 5 remote machines.
3. Buses shall be identified by name rather than number. Bus name shall include bus voltage base.

### 2.18.7 Additional Resources

Below are some great resources if you're looking to learn more about power flow, transient stability, and other subjects related to this project.

- Kundur, Prabha. *Power System Stability and Control*. McGraw-Hill, Inc. 1994
- <http://home.engineering.iastate.edu/~jdm/ee553/Tinney1.pdf>

## C

cf\_debug (C++ member), 409  
CFLOW\_IPC\_BUFF\_SIZE (C macro), 389

## E

err\_buf (C++ member), 409

## I

input\_data (C++ union), 348  
input\_data::ACbus (C++ member), 348  
input\_data::area (C++ member), 349  
input\_data::branch (C++ member), 348  
input\_data::cbus (C++ member), 349  
input\_data::cmnt (C++ member), 349  
input\_data::DCbus (C++ member), 348  
input\_data::E (C++ member), 348  
input\_data::itie (C++ member), 349  
input\_data::L (C++ member), 348  
input\_data::LD (C++ member), 348  
input\_data::LM (C++ member), 348  
input\_data::qcurve (C++ member), 349  
input\_data::R (C++ member), 349  
input\_data::RM (C++ member), 349  
input\_data::RN (C++ member), 349  
input\_data::RP (C++ member), 349  
input\_data::RQ (C++ member), 349  
input\_data::RV (C++ member), 349  
input\_data::RZ (C++ member), 349  
input\_data::T (C++ member), 349  
input\_data::TP (C++ member), 349  
input\_data::xdata (C++ member), 349

## P

pf\_AC\_bus (C++ struct), 350  
pf\_AC\_bus::dummy1 (C++ member), 350  
pf\_AC\_bus::dummy2 (C++ member), 351  
pf\_AC\_bus::kv (C++ member), 350  
pf\_AC\_bus::name (C++ member), 350  
pf\_AC\_bus::owner (C++ member), 350  
pf\_AC\_bus::pct\_vars (C++ member), 351  
pf\_AC\_bus::Pgen (C++ member), 350  
pf\_AC\_bus::Pload (C++ member), 350

pf\_AC\_bus::Pmax (C++ member), 350  
pf\_AC\_bus::Pshunt (C++ member), 350  
pf\_AC\_bus::Qload (C++ member), 350  
pf\_AC\_bus::Qmin (C++ member), 351  
pf\_AC\_bus::Qsch\_Qmax (C++ member), 350  
pf\_AC\_bus::Qshunt (C++ member), 350  
pf\_AC\_bus::rmt\_kv (C++ member), 351  
pf\_AC\_bus::rmt\_name (C++ member), 351  
pf\_AC\_bus::type (C++ member), 350  
pf\_AC\_bus::Vhold\_Vmax (C++ member), 351  
pf\_AC\_bus::Vmin\_Vdeg (C++ member), 351  
pf\_AC\_bus::zone (C++ member), 350  
pf\_area (C++ struct), 351  
pf\_area::max\_Vpu (C++ member), 352  
pf\_area::min\_Vpu (C++ member), 352  
pf\_area::name (C++ member), 351  
pf\_area::sbus\_kv (C++ member), 351  
pf\_area::sbus\_name (C++ member), 351  
pf\_area::sched\_export (C++ member), 351  
pf\_area::type (C++ member), 351  
pf\_area::zone0 (C++ member), 352  
pf\_area::zone1 (C++ member), 352  
pf\_area::zone2 (C++ member), 352  
pf\_area::zone3 (C++ member), 352  
pf\_area::zone4 (C++ member), 352  
pf\_area::zone5 (C++ member), 352  
pf\_area::zone6 (C++ member), 352  
pf\_area::zone7 (C++ member), 352  
pf\_area::zone8 (C++ member), 352  
pf\_area::zone9 (C++ member), 352  
pf\_area\_of\_zone (C++ function), 399  
pf\_area\_soln (C++ struct), 352  
pf\_area\_soln::Pexport (C++ member), 353  
pf\_area\_soln::Pgen (C++ member), 353  
pf\_area\_soln::Pload (C++ member), 353  
pf\_area\_soln::Ploss (C++ member), 353  
pf\_area\_soln::type (C++ member), 353  
pf\_branch (C++ struct), 353  
pf\_branch::alpha\_N\_deg (C++ member), 354  
pf\_branch::b (C++ member), 354  
pf\_branch::bottleneck\_rating (C++ member), 355  
pf\_branch::bus1\_kv (C++ member), 353

pf\_branch::bus1\_name (C++ member), 353  
 pf\_branch::bus2\_kv (C++ member), 353  
 pf\_branch::bus2\_name (C++ member), 353  
 pf\_branch::ckt\_id (C++ member), 354  
 pf\_branch::date\_in (C++ member), 354  
 pf\_branch::date\_out (C++ member), 355  
 pf\_branch::descrip (C++ member), 354  
 pf\_branch::emergency\_rating (C++ member), 355  
 pf\_branch::g (C++ member), 354  
 pf\_branch::gamma\_0\_deg (C++ member), 354  
 pf\_branch::meter (C++ member), 353  
 pf\_branch::num\_ckts (C++ member), 354  
 pf\_branch::owner (C++ member), 353  
 pf\_branch::r (C++ member), 354  
 pf\_branch::section (C++ member), 354  
 pf\_branch::tap1 (C++ member), 354  
 pf\_branch::tap2 (C++ member), 354  
 pf\_branch::thermal\_rating (C++ member), 355  
 pf\_branch::total\_rating (C++ member), 354  
 pf\_branch::type (C++ member), 353  
 pf\_branch::x (C++ member), 354  
 pf\_branch\_soln (C++ struct), 355  
 pf\_branch\_soln::crit\_line\_load\_amps (C++ member), 356  
 pf\_branch\_soln::crit\_line\_load\_term (C++ member), 356  
 pf\_branch\_soln::crit\_line\_rat\_amps (C++ member), 356  
 pf\_branch\_soln::crit\_line\_rat\_code (C++ member), 356  
 pf\_branch\_soln::crit\_xfmr\_load\_mva (C++ member), 356  
 pf\_branch\_soln::crit\_xfmr\_load\_term (C++ member), 356  
 pf\_branch\_soln::crit\_xfmr\_rat\_code (C++ member), 356  
 pf\_branch\_soln::crit\_xfmr\_rat\_mva (C++ member), 356  
 pf\_branch\_soln::num\_ckts (C++ member), 355  
 pf\_branch\_soln::Pin (C++ member), 355  
 pf\_branch\_soln::Ploss (C++ member), 355  
 pf\_branch\_soln::Pout (C++ member), 355  
 pf\_branch\_soln::Qin (C++ member), 355  
 pf\_branch\_soln::Qloss (C++ member), 355  
 pf\_branch\_soln::Qout (C++ member), 355  
 pf\_branch\_soln::tap1 (C++ member), 356  
 pf\_branch\_soln::tap2 (C++ member), 356  
 pf\_branch\_soln::tot\_line\_load\_amps (C++ member), 356  
 pf\_branch\_soln::tot\_line\_load\_pct (C++ member), 356  
 pf\_branch\_soln::tot\_xfmr\_load\_mva (C++ member), 356  
 pf\_branch\_soln::tot\_xfmr\_load\_pct (C++ member), 356  
 pf\_branch\_soln::type (C++ member), 355  
 pf\_bus\_AC\_soln (C++ struct), 356  
 pf\_bus\_AC\_soln::Bshunt\_sch (C++ member), 357  
 pf\_bus\_AC\_soln::Bshunt\_sch\_cap (C++ member), 357  
 pf\_bus\_AC\_soln::Bshunt\_sch\_rx (C++ member), 357  
 pf\_bus\_AC\_soln::Bshunt\_used (C++ member), 357  
 pf\_bus\_AC\_soln::Bshunt\_used\_cap (C++ member), 357  
 pf\_bus\_AC\_soln::Bshunt\_used\_rx (C++ member), 357  
 pf\_bus\_AC\_soln::Pgen (C++ member), 357  
 pf\_bus\_AC\_soln::Pload (C++ member), 357  
 pf\_bus\_AC\_soln::Qgen (C++ member), 357  
 pf\_bus\_AC\_soln::Qload (C++ member), 357  
 pf\_bus\_AC\_soln::Qunsch (C++ member), 357  
 pf\_bus\_AC\_soln::type (C++ member), 357  
 pf\_bus\_AC\_soln::Vdeg (C++ member), 357  
 pf\_bus\_AC\_soln::Vmag (C++ member), 357  
 pf\_bus\_DC\_soln (C++ struct), 358  
 pf\_bus\_DC\_soln::converter\_deg (C++ member), 358  
 pf\_bus\_DC\_soln::dummy1 (C++ member), 358  
 pf\_bus\_DC\_soln::dummy2 (C++ member), 358  
 pf\_bus\_DC\_soln::dummy3 (C++ member), 358  
 pf\_bus\_DC\_soln::dummy4 (C++ member), 358  
 pf\_bus\_DC\_soln::dummy5 (C++ member), 358  
 pf\_bus\_DC\_soln::dummy6 (C++ member), 358  
 pf\_bus\_DC\_soln::dummy7 (C++ member), 359  
 pf\_bus\_DC\_soln::P\_DC (C++ member), 358  
 pf\_bus\_DC\_soln::P\_valve\_losses (C++ member), 358  
 pf\_bus\_DC\_soln::Q\_DC (C++ member), 358  
 pf\_bus\_DC\_soln::Q\_valve\_losses (C++ member), 358  
 pf\_bus\_DC\_soln::type (C++ member), 358  
 pf\_bus\_DC\_soln::V\_DC (C++ member), 358  
 pf\_bus\_exists (C++ function), 401  
 pf\_case\_info (C++ function), 399  
 pf\_case\_stats (C++ struct), 359  
 pf\_case\_stats::base\_mva (C++ member), 359  
 pf\_case\_stats::case\_soln\_status (C++ member), 360  
 pf\_case\_stats::num\_adjustable\_buses (C++ member), 360  
 pf\_case\_stats::num\_AGC\_buses (C++ member), 359  
 pf\_case\_stats::num\_area\_slack\_buses (C++ member), 359  
 pf\_case\_stats::num\_areas (C++ member), 359  
 pf\_case\_stats::num\_branches (C++ member), 360  
 pf\_case\_stats::num\_buses (C++ member), 359

pf\_case\_stats::num\_BX\_buses (C++ member), 359  
 pf\_case\_stats::num\_circuits (C++ member), 360  
 pf\_case\_stats::num\_DC\_buses (C++ member), 359  
 pf\_case\_stats::num\_DC\_lines (C++ member), 360  
 pf\_case\_stats::num\_DC\_systems (C++ member), 359  
 pf\_case\_stats::num\_diff\_kv (C++ member), 360  
 pf\_case\_stats::num\_ities (C++ member), 359  
 pf\_case\_stats::num\_LTC\_xfmrs (C++ member), 360  
 pf\_case\_stats::num\_owners (C++ member), 359  
 pf\_case\_stats::num\_pct\_var\_ctrl\_buses (C++ member), 360  
 pf\_case\_stats::num\_phase\_shifters (C++ member), 360  
 pf\_case\_stats::num\_rec\_types (C++ member), 360  
 pf\_case\_stats::num\_zones (C++ member), 359  
 pf\_case\_stats::PF\_version (C++ member), 359  
 pf\_cbus (C++ struct), 360  
 pf\_cbus::Bshunt (C++ member), 361  
 pf\_cbus::code\_year (C++ member), 361  
 pf\_cbus::Gshunt (C++ member), 361  
 pf\_cbus::kv (C++ member), 361  
 pf\_cbus::name (C++ member), 361  
 pf\_cbus::owner (C++ member), 361  
 pf\_cbus::Pgen (C++ member), 361  
 pf\_cbus::Pload (C++ member), 361  
 pf\_cbus::Qgen\_Qmax (C++ member), 361  
 pf\_cbus::Qload (C++ member), 361  
 pf\_cbus::Qmin (C++ member), 361  
 pf\_cbus::type (C++ member), 361  
 pf\_cbus\_soln (C++ struct), 361  
 pf\_cbus\_soln::Bshunt (C++ member), 362  
 pf\_cbus\_soln::Gshunt (C++ member), 362  
 pf\_cbus\_soln::Pgen (C++ member), 362  
 pf\_cbus\_soln::Pload (C++ member), 362  
 pf\_cbus\_soln::Qgen (C++ member), 362  
 pf\_cbus\_soln::Qload (C++ member), 362  
 pf\_cbus\_soln::type (C++ member), 362  
 pf\_cflow\_exit (C++ function), 390  
 pf\_cflow\_inbuf (C++ member), 409  
 pf\_cflow\_init (C++ function), 390  
 pf\_cflow\_ipc (C++ function), 390  
 pf\_cflow\_outbuf (C++ member), 409  
 pf\_cflow\_socket (C++ member), 409  
 pf\_command (C++ function), 409  
 pf\_comments (C++ struct), 362  
 pf\_comments::c (C++ member), 362  
 pf\_comments::case\_desc (C++ member), 362  
 pf\_comments::case\_name (C++ member), 362  
 pf\_comments::h (C++ member), 362  
 pf\_DC\_bus (C++ struct), 362  
 pf\_DC\_bus::alpha\_gamma\_N\_deg (C++ member), 363  
 pf\_DC\_bus::alpha\_min\_deg (C++ member), 363  
 pf\_DC\_bus::alpha\_stop\_deg (C++ member), 363  
 pf\_DC\_bus::bridge\_current\_rating\_amps (C++ member), 363  
 pf\_DC\_bus::bridges\_per\_ckt (C++ member), 363  
 pf\_DC\_bus::commutating\_bus\_kv (C++ member), 364  
 pf\_DC\_bus::commutating\_bus\_name (C++ member), 364  
 pf\_DC\_bus::converter\_code (C++ member), 364  
 pf\_DC\_bus::dummy1 (C++ member), 363  
 pf\_DC\_bus::dummy2 (C++ member), 364  
 pf\_DC\_bus::gamma\_0\_deg (C++ member), 363  
 pf\_DC\_bus::kv (C++ member), 363  
 pf\_DC\_bus::name (C++ member), 363  
 pf\_DC\_bus::owner (C++ member), 363  
 pf\_DC\_bus::P\_sched (C++ member), 363  
 pf\_DC\_bus::smooth\_rx\_mh (C++ member), 363  
 pf\_DC\_bus::type (C++ member), 363  
 pf\_DC\_bus::V\_sched (C++ member), 363  
 pf\_DC\_bus::valve\_drop\_per\_bridge\_volts (C++ member), 363  
 pf\_DC\_bus::zone (C++ member), 363  
 pf\_del\_area (C++ function), 390  
 pf\_del\_zone (C++ function), 390  
 pf\_E (C++ struct), 364  
 pf\_E::b1 (C++ member), 365  
 pf\_E::b2 (C++ member), 365  
 pf\_E::bottleneck\_rating (C++ member), 366  
 pf\_E::bus1\_kv (C++ member), 364  
 pf\_E::bus1\_name (C++ member), 364  
 pf\_E::bus2\_kv (C++ member), 364  
 pf\_E::bus2\_name (C++ member), 364  
 pf\_E::ckt\_id (C++ member), 365  
 pf\_E::date\_in (C++ member), 365  
 pf\_E::date\_out (C++ member), 366  
 pf\_E::dummy1 (C++ member), 365  
 pf\_E::dummy2 (C++ member), 365  
 pf\_E::dummy3 (C++ member), 365  
 pf\_E::dummy4 (C++ member), 366  
 pf\_E::g1 (C++ member), 365  
 pf\_E::g2 (C++ member), 365  
 pf\_E::meter (C++ member), 364  
 pf\_E::num\_ckts (C++ member), 365  
 pf\_E::owner (C++ member), 364  
 pf\_E::r (C++ member), 365  
 pf\_E::section (C++ member), 365  
 pf\_E::thermal\_rating (C++ member), 366  
 pf\_E::total\_rating (C++ member), 365  
 pf\_E::type (C++ member), 364  
 pf\_E::x (C++ member), 365  
 pf\_get\_list (C++ function), 398  
 pf\_init (C++ function), 398  
 pf\_init\_area (C++ function), 401  
 pf\_init\_branch (C++ function), 401  
 pf\_init\_bus (C++ function), 400



pf\_init\_cbus (C++ function), 400  
 pf\_init\_itie (C++ function), 400  
 pf\_init\_qcurve (C++ function), 399  
 pf\_init\_rec (C++ function), 399  
 pf\_itie (C++ struct), 366  
 pf\_itie::areal\_name (C++ member), 366  
 pf\_itie::area2\_name (C++ member), 366  
 pf\_itie::sched\_export (C++ member), 366  
 pf\_itie::type (C++ member), 366  
 pf\_itie\_soln (C++ struct), 366  
 pf\_itie\_soln::input\_exists (C++ member), 367  
 pf\_itie\_soln::Pcirc (C++ member), 367  
 pf\_itie\_soln::Pexport (C++ member), 367  
 pf\_itie\_soln::type (C++ member), 367  
 pf\_L (C++ struct), 367  
 pf\_L::b (C++ member), 368  
 pf\_L::bottleneck\_rating (C++ member), 369  
 pf\_L::bus1\_kv (C++ member), 367  
 pf\_L::bus1\_name (C++ member), 367  
 pf\_L::bus2\_kv (C++ member), 367  
 pf\_L::bus2\_name (C++ member), 367  
 pf\_L::ckt\_id (C++ member), 368  
 pf\_L::date\_in (C++ member), 368  
 pf\_L::date\_out (C++ member), 368  
 pf\_L::descrip (C++ member), 368  
 pf\_L::dummy1 (C++ member), 368  
 pf\_L::dummy2 (C++ member), 368  
 pf\_L::dummy3 (C++ member), 368  
 pf\_L::dummy4 (C++ member), 369  
 pf\_L::g (C++ member), 368  
 pf\_L::meter (C++ member), 367  
 pf\_L::miles (C++ member), 368  
 pf\_L::num\_ckts (C++ member), 368  
 pf\_L::owner (C++ member), 367  
 pf\_L::r (C++ member), 368  
 pf\_L::section (C++ member), 368  
 pf\_L::thermal\_rating (C++ member), 369  
 pf\_L::total\_rating (C++ member), 368  
 pf\_L::type (C++ member), 367  
 pf\_L::x (C++ member), 368  
 pf\_LD (C++ struct), 369  
 pf\_LD::alpha\_N\_deg (C++ member), 370  
 pf\_LD::bottleneck\_rating (C++ member), 371  
 pf\_LD::bus1\_kv (C++ member), 369  
 pf\_LD::bus1\_name (C++ member), 369  
 pf\_LD::bus2\_kv (C++ member), 369  
 pf\_LD::bus2\_name (C++ member), 369  
 pf\_LD::C\_uf (C++ member), 370  
 pf\_LD::dummy1 (C++ member), 370  
 pf\_LD::dummy2 (C++ member), 370  
 pf\_LD::dummy3 (C++ member), 370  
 pf\_LD::dummy4 (C++ member), 370  
 pf\_LD::dummy5 (C++ member), 370  
 pf\_LD::dummy6 (C++ member), 371  
 pf\_LD::gamma\_0\_deg (C++ member), 370  
 pf\_LD::I\_or\_R\_control (C++ member), 369  
 pf\_LD::L\_mh (C++ member), 370  
 pf\_LD::meter (C++ member), 369  
 pf\_LD::miles (C++ member), 370  
 pf\_LD::owner (C++ member), 369  
 pf\_LD::P\_sched (C++ member), 370  
 pf\_LD::R (C++ member), 370  
 pf\_LD::thermal\_rating (C++ member), 370  
 pf\_LD::total\_rating (C++ member), 370  
 pf\_LD::type (C++ member), 369  
 pf\_LD::V\_sched (C++ member), 370  
 pf\_list\_type (C++ enum), 389  
 pf\_list\_type::AREA\_LIST (C++ enumerator), 389  
 pf\_list\_type::BUS\_LIST (C++ enumerator), 389  
 pf\_list\_type::KV\_LIST (C++ enumerator), 389  
 pf\_list\_type::OWNER\_LIST (C++ enumerator), 389  
 pf\_list\_type::REC\_TYPE\_LIST (C++ enumerator), 389  
 pf\_list\_type::ZONE\_LIST (C++ enumerator), 389  
 pf\_LM (C++ struct), 371  
 pf\_LM::bottleneck\_rating (C++ member), 372  
 pf\_LM::bus1\_kv (C++ member), 371  
 pf\_LM::bus1\_name (C++ member), 371  
 pf\_LM::bus2\_kv (C++ member), 371  
 pf\_LM::bus2\_name (C++ member), 371  
 pf\_LM::C\_uf (C++ member), 372  
 pf\_LM::date\_in (C++ member), 372  
 pf\_LM::date\_out (C++ member), 372  
 pf\_LM::dummy1 (C++ member), 371  
 pf\_LM::dummy2 (C++ member), 371  
 pf\_LM::dummy3 (C++ member), 372  
 pf\_LM::dummy4 (C++ member), 372  
 pf\_LM::dummy5 (C++ member), 372  
 pf\_LM::dummy6 (C++ member), 372  
 pf\_LM::dummy7 (C++ member), 372  
 pf\_LM::dummy8 (C++ member), 372  
 pf\_LM::dummy9 (C++ member), 373  
 pf\_LM::L\_mh (C++ member), 372  
 pf\_LM::meter (C++ member), 371  
 pf\_LM::miles (C++ member), 372  
 pf\_LM::owner (C++ member), 371  
 pf\_LM::R (C++ member), 372  
 pf\_LM::thermal\_rating (C++ member), 372  
 pf\_LM::total\_rating (C++ member), 372  
 pf\_LM::type (C++ member), 371  
 pf\_load\_changes (C++ function), 392  
 pf\_load\_netdata (C++ function), 392  
 pf\_load\_oldbase (C++ function), 392  
 pf\_load\_refbase (C++ function), 408  
 pf\_plot (C++ function), 407  
 pf\_put\_inrec (C++ function), 392  
 pf\_qcurve (C++ struct), 373  
 pf\_qcurve::active (C++ member), 373



pf\_qcurve::bus\_kv (C++ member), 373  
 pf\_qcurve::bus\_name (C++ member), 373  
 pf\_qcurve::Pgen0 (C++ member), 373  
 pf\_qcurve::Pgen1 (C++ member), 373  
 pf\_qcurve::Pgen2 (C++ member), 373  
 pf\_qcurve::Pgen3 (C++ member), 373  
 pf\_qcurve::Pgen4 (C++ member), 373  
 pf\_qcurve::Pgen5 (C++ member), 373  
 pf\_qcurve::Pgen6 (C++ member), 373  
 pf\_qcurve::Pgen7 (C++ member), 374  
 pf\_qcurve::Pgen8 (C++ member), 374  
 pf\_qcurve::Pgen9 (C++ member), 374  
 pf\_qcurve::PU\_code (C++ member), 373  
 pf\_qcurve::type (C++ member), 373  
 pf\_qcurve\_soln (C++ struct), 374  
 pf\_qcurve\_soln::Pgen (C++ member), 374  
 pf\_qcurve\_soln::Qgen (C++ member), 374  
 pf\_qcurve\_soln::type (C++ member), 374  
 pf\_R (C++ struct), 374  
 pf\_R::bus1\_kv (C++ member), 374  
 pf\_R::bus1\_name (C++ member), 374  
 pf\_R::bus2\_kv (C++ member), 375  
 pf\_R::bus2\_name (C++ member), 375  
 pf\_R::date\_in (C++ member), 376  
 pf\_R::date\_out (C++ member), 376  
 pf\_R::dummy1 (C++ member), 375  
 pf\_R::dummy2 (C++ member), 375  
 pf\_R::dummy3 (C++ member), 375  
 pf\_R::dummy4 (C++ member), 375  
 pf\_R::dummy5 (C++ member), 375  
 pf\_R::dummy6 (C++ member), 375  
 pf\_R::dummy7 (C++ member), 375  
 pf\_R::dummy8 (C++ member), 375  
 pf\_R::dummy9 (C++ member), 375  
 pf\_R::max\_tap (C++ member), 375  
 pf\_R::min\_tap (C++ member), 375  
 pf\_R::num\_taps (C++ member), 375  
 pf\_R::owner (C++ member), 374  
 pf\_R::Qmax (C++ member), 376  
 pf\_R::Qmin (C++ member), 376  
 pf\_R::rmt\_bus\_kv (C++ member), 376  
 pf\_R::rmt\_bus\_name (C++ member), 376  
 pf\_R::type (C++ member), 374  
 pf\_R::var\_tap\_side (C++ member), 374  
 pf\_rec (C++ struct), 376  
 pf\_rec::i (C++ member), 376  
 pf\_rec::s (C++ member), 376  
 pf\_rec\_a2b (C++ function), 397  
 pf\_rec\_area (C++ function), 393  
 pf\_rec\_b2a (C++ function), 397  
 pf\_rec\_branch (C++ function), 393  
 pf\_rec\_bus (C++ function), 394  
 pf\_rec\_cbus (C++ function), 395  
 pf\_rec\_comments (C++ function), 395  
 pf\_rec\_itie (C++ function), 395  
 pf\_rec\_qcurve (C++ function), 396  
 pf\_rec\_xdata (C++ function), 396  
 pf\_rename\_area (C++ function), 391  
 pf\_rename\_bus (C++ function), 391  
 pf\_rename\_zone (C++ function), 391  
 pf\_RM (C++ struct), 376  
 pf\_RM::bus1\_kv (C++ member), 377  
 pf\_RM::bus1\_name (C++ member), 377  
 pf\_RM::bus2\_kv (C++ member), 377  
 pf\_RM::bus2\_name (C++ member), 377  
 pf\_RM::date\_in (C++ member), 378  
 pf\_RM::date\_out (C++ member), 378  
 pf\_RM::dummy1 (C++ member), 377  
 pf\_RM::dummy2 (C++ member), 377  
 pf\_RM::dummy3 (C++ member), 377  
 pf\_RM::dummy4 (C++ member), 377  
 pf\_RM::dummy5 (C++ member), 377  
 pf\_RM::dummy6 (C++ member), 377  
 pf\_RM::dummy7 (C++ member), 378  
 pf\_RM::dummy8 (C++ member), 378  
 pf\_RM::dummy9 (C++ member), 378  
 pf\_RM::max\_phase\_shift\_deg (C++ member), 378  
 pf\_RM::min\_phase\_shift\_deg (C++ member), 378  
 pf\_RM::num\_taps (C++ member), 377  
 pf\_RM::owner (C++ member), 377  
 pf\_RM::Pmax (C++ member), 378  
 pf\_RM::Pmin (C++ member), 378  
 pf\_RM::rmt\_bus\_kv (C++ member), 378  
 pf\_RM::rmt\_bus\_name (C++ member), 378  
 pf\_RM::type (C++ member), 377  
 pf\_RM::var\_tap\_side (C++ member), 377  
 pf\_RN (C++ type), 389  
 pf\_RP (C++ type), 389  
 pf\_RQ (C++ type), 389  
 pf\_RV (C++ type), 389  
 pf\_RZ (C++ struct), 378  
 pf\_RZ::Bis\_max (C++ member), 380  
 pf\_RZ::Bis\_min (C++ member), 380  
 pf\_RZ::bus1\_kv (C++ member), 379  
 pf\_RZ::bus1\_name (C++ member), 379  
 pf\_RZ::bus2\_kv (C++ member), 379  
 pf\_RZ::bus2\_name (C++ member), 379  
 pf\_RZ::ckt\_id (C++ member), 379  
 pf\_RZ::dummy1 (C++ member), 380  
 pf\_RZ::dummy2 (C++ member), 380  
 pf\_RZ::dummy3 (C++ member), 380  
 pf\_RZ::dummy4 (C++ member), 380  
 pf\_RZ::dummy5 (C++ member), 380  
 pf\_RZ::dummy6 (C++ member), 380  
 pf\_RZ::dummy7 (C++ member), 380  
 pf\_RZ::dummy8 (C++ member), 380  
 pf\_RZ::I\_rate (C++ member), 379  
 pf\_RZ::owner (C++ member), 379

pf\_RZ::Pc\_max (C++ member), 379  
 pf\_RZ::Pc\_min (C++ member), 379  
 pf\_RZ::rani\_type (C++ member), 379  
 pf\_RZ::section (C++ member), 379  
 pf\_RZ::type (C++ member), 379  
 pf\_RZ::var\_tap\_side (C++ member), 379  
 pf\_RZ::Xij\_max (C++ member), 380  
 pf\_RZ::Xij\_min (C++ member), 380  
 pf\_save\_changes (C++ function), 397  
 pf\_save\_netdata (C++ function), 397  
 pf\_save\_newbase (C++ function), 398  
 pf\_save\_wscs\_stab\_data (C++ function), 398  
 pf\_select\_base (C++ function), 408  
 pf\_solution (C++ function), 398  
 pf\_solve\_area (C++ function), 408  
 pf\_T (C++ struct), 380  
 pf\_T::b (C++ member), 382  
 pf\_T::bottleneck\_rating (C++ member), 382  
 pf\_T::bus1\_kv (C++ member), 381  
 pf\_T::bus1\_name (C++ member), 381  
 pf\_T::bus2\_kv (C++ member), 381  
 pf\_T::bus2\_name (C++ member), 381  
 pf\_T::ckt\_id (C++ member), 381  
 pf\_T::date\_in (C++ member), 382  
 pf\_T::date\_out (C++ member), 382  
 pf\_T::dummy1 (C++ member), 382  
 pf\_T::dummy2 (C++ member), 382  
 pf\_T::dummy3 (C++ member), 382  
 pf\_T::emergency\_rating (C++ member), 382  
 pf\_T::g (C++ member), 381  
 pf\_T::meter (C++ member), 381  
 pf\_T::num\_ckts (C++ member), 381  
 pf\_T::owner (C++ member), 381  
 pf\_T::r (C++ member), 381  
 pf\_T::section (C++ member), 381  
 pf\_T::tap1 (C++ member), 382  
 pf\_T::tap2 (C++ member), 382  
 pf\_T::thermal\_rating (C++ member), 382  
 pf\_T::total\_rating (C++ member), 381  
 pf\_T::type (C++ member), 381  
 pf\_T::x (C++ member), 381  
 pf\_TP (C++ struct), 382  
 pf\_TP::b (C++ member), 384  
 pf\_TP::bottleneck\_rating (C++ member), 384  
 pf\_TP::bus1\_kv (C++ member), 383  
 pf\_TP::bus1\_name (C++ member), 383  
 pf\_TP::bus2\_kv (C++ member), 383  
 pf\_TP::bus2\_name (C++ member), 383  
 pf\_TP::ckt\_id (C++ member), 383  
 pf\_TP::date\_in (C++ member), 384  
 pf\_TP::date\_out (C++ member), 384  
 pf\_TP::dummy1 (C++ member), 384  
 pf\_TP::dummy2 (C++ member), 384  
 pf\_TP::dummy3 (C++ member), 384  
 pf\_TP::emergency\_rating (C++ member), 384  
 pf\_TP::g (C++ member), 383  
 pf\_TP::meter (C++ member), 383  
 pf\_TP::num\_ckts (C++ member), 383  
 pf\_TP::owner (C++ member), 383  
 pf\_TP::phase\_shift\_deg (C++ member), 384  
 pf\_TP::r (C++ member), 383  
 pf\_TP::section (C++ member), 383  
 pf\_TP::tap2 (C++ member), 384  
 pf\_TP::thermal\_rating (C++ member), 384  
 pf\_TP::total\_rating (C++ member), 383  
 pf\_TP::type (C++ member), 383  
 pf\_TP::x (C++ member), 383  
 pf\_user\_branch (C++ function), 405  
 pf\_user\_bus (C++ function), 406  
 pf\_user\_comment (C++ function), 404  
 pf\_user\_define (C++ function), 403  
 pf\_user\_init\_def (C++ function), 402  
 pf\_user\_itie (C++ function), 406  
 pf\_user\_load\_def (C++ function), 402  
 pf\_user\_quantity (C++ function), 404  
 pf\_user\_report (C++ function), 403  
 pf\_user\_string (C++ function), 405  
 pf\_user\_sub\_def (C++ function), 402  
 pf\_xdata (C++ struct), 384  
 pf\_xdata::bus\_kv (C++ member), 385  
 pf\_xdata::bus\_name (C++ member), 385  
 pf\_xdata::owner (C++ member), 385  
 pf\_xdata::rmt\_kv (C++ member), 385  
 pf\_xdata::rmt\_name (C++ member), 385  
 pf\_xdata::seg1\_delta\_mva (C++ member), 385  
 pf\_xdata::seg1\_num\_steps (C++ member), 385  
 pf\_xdata::seg2\_delta\_mva (C++ member), 385  
 pf\_xdata::seg2\_num\_steps (C++ member), 385  
 pf\_xdata::seg3\_delta\_mva (C++ member), 385  
 pf\_xdata::seg3\_num\_steps (C++ member), 385  
 pf\_xdata::seg4\_delta\_mva (C++ member), 385  
 pf\_xdata::seg4\_num\_steps (C++ member), 385  
 pf\_xdata::seg5\_delta\_mva (C++ member), 385  
 pf\_xdata::seg5\_num\_steps (C++ member), 385  
 pf\_xdata::seg6\_delta\_mva (C++ member), 386  
 pf\_xdata::seg6\_num\_steps (C++ member), 386  
 pf\_xdata::seg7\_delta\_mva (C++ member), 386  
 pf\_xdata::seg7\_num\_steps (C++ member), 386  
 pf\_xdata::seg8\_delta\_mva (C++ member), 386  
 pf\_xdata::seg8\_num\_steps (C++ member), 386  
 pf\_xdata::type (C++ member), 385  
 pf\_xdata\_soln (C++ struct), 386  
 pf\_xdata\_soln::seg1\_mvar\_per\_unit (C++ member), 386  
 pf\_xdata\_soln::seg1\_sch\_units (C++ member), 386  
 pf\_xdata\_soln::seg1\_used\_units (C++ member), 386

pf\_xdata\_soln::seg2\_mvar\_per\_unit (C++ member), 386  
 pf\_xdata\_soln::seg2\_sch\_units (C++ member), 386  
 pf\_xdata\_soln::seg2\_used\_units (C++ member), 386  
 pf\_xdata\_soln::seg3\_mvar\_per\_unit (C++ member), 387  
 pf\_xdata\_soln::seg3\_sch\_units (C++ member), 387  
 pf\_xdata\_soln::seg3\_used\_units (C++ member), 387  
 pf\_xdata\_soln::seg4\_mvar\_per\_unit (C++ member), 387  
 pf\_xdata\_soln::seg4\_sch\_units (C++ member), 387  
 pf\_xdata\_soln::seg4\_used\_units (C++ member), 387  
 pf\_xdata\_soln::seg5\_mvar\_per\_unit (C++ member), 387  
 pf\_xdata\_soln::seg5\_sch\_units (C++ member), 387  
 pf\_xdata\_soln::seg5\_used\_units (C++ member), 387  
 pf\_xdata\_soln::seg6\_mvar\_per\_unit (C++ member), 387  
 pf\_xdata\_soln::seg6\_sch\_units (C++ member), 387  
 pf\_xdata\_soln::seg6\_used\_units (C++ member), 387  
 pf\_xdata\_soln::seg7\_mvar\_per\_unit (C++ member), 387  
 pf\_xdata\_soln::seg7\_sch\_units (C++ member), 387  
 pf\_xdata\_soln::seg7\_used\_units (C++ member), 387  
 pf\_xdata\_soln::seg8\_mvar\_per\_unit (C++ member), 388  
 pf\_xdata\_soln::seg8\_sch\_units (C++ member), 388  
 pf\_xdata\_soln::seg8\_used\_units (C++ member), 388  
 pf\_xdata\_soln::type (C++ member), 386  
 solution\_data::itie (C++ member), 388  
 solution\_data::qcurve (C++ member), 388  
 solution\_data::xdata (C++ member), 388

## R

reply\_pf (C++ member), 409

## S

solution\_data (C++ union), 388  
 solution\_data::ACbus (C++ member), 388  
 solution\_data::area (C++ member), 388  
 solution\_data::branch (C++ member), 388  
 solution\_data::cbus (C++ member), 388  
 solution\_data::DCbus (C++ member), 388