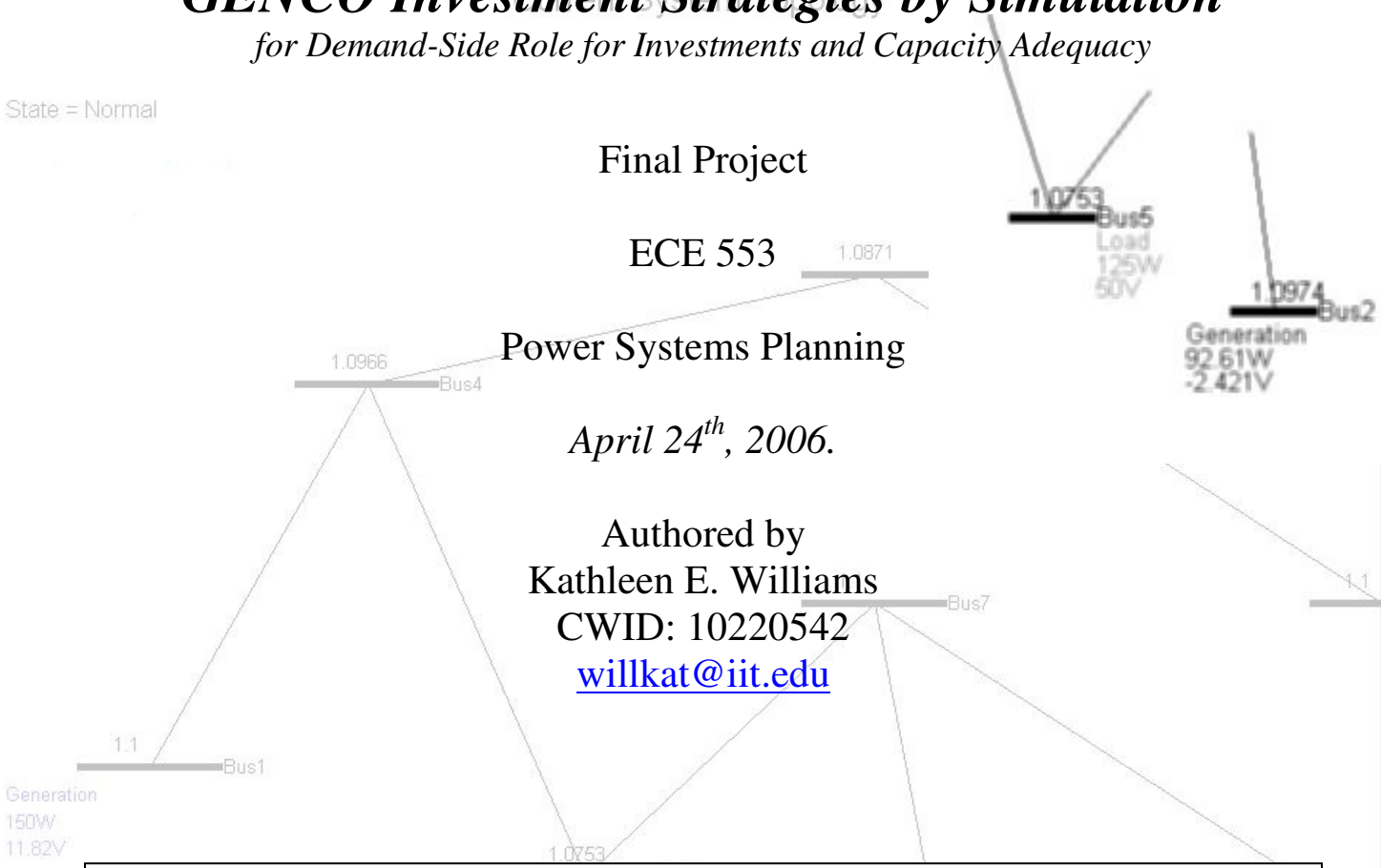
GENCO Investment Strategies by Simulation

for Demand-Side Role for Investments and Capacity Adequacy



Final Project

ECE 553

Power Systems Planning

April 24th, 2006.

Authored by
Kathleen E. Williams
CWID: 10220542
willkat@iit.edu

Abstract:

This project will present an applied and game-like approach to simulating the load growth, investment decisions by two types of generation technologies, demand-price responsiveness, and reliability, of a test-case power system. The simulation begins as a 9-bus system with existing generation (3 generators) and transmission lines (8 lines). System topology can be viewed in a figure throughout the game with the yearly generation and load at each bus. In addition, dynamic color-coding is used to highlight transmission lines that exceed MVA ratings and highlight bus voltages that violate any limits. The winning objective of the player company (you) is to maximize his profit. Reliability can be tracked by viewing the N-1 generator and line contingencies every year, but this does not influence profits. There are two generation technologies used: coal and gas turbine. Each technology will have a similar competitor in the simulation. The competitor can bring down the market price and reduce the player's profits significantly. The clock starts at T=0 in the investment game with a historical record of past prices and projected prices based on lack of investment. As time moves forward in yearly increments, the load, prices, investment costs, and other variables are adjusted to that of the player's performance. The player has the opportunity to study various profitable and unprofitable investment alternatives each year of the simulation. If he invests at the right location, and in the right planning year, his company can make windfall profits. Competitors randomly participate in adding extra generation in random areas of the system based on the competition level settings. The challenge for the user is to study the effects of his investment decisions on market prices, reliability, and his profitability.

Table of Contents

I. Introduction

There are two extreme options for ensuring adequate generation supply and maintaining reliability: the energy-only market option and the regulatory authority/ISO-based option.
Alex D. Papalexopoulos - "Supplying the Generation to Meet the Demand"

Background

The demand for cost efficiency, which has caused an upsurge of deregulation and liberalization initiatives in the power industry, will play a key role in current and future market designs. The objective behind power system deregulation is to increase the competition, and with that the economic efficiency in the building and operation of the electrical power system. Liberalized markets focus on profit maximization in lieu of cost minimization as under regulation. Uncertainty in the power market spot prices and decentralized decisions bare more risk on the investors than under the traditional regulation where utilities were allowed to recover their risk under rates. A common challenge amongst market designers and policy makers is how to keep a competitive energy market that can ensure sufficient generation supply to meet the demand and ensure reliability.

Although capacity in the electricity infrastructure was adequate for 2005, many still see resource adequacy as a growing concern because rate of investment in generation and transmission is seen by some as too low to meet future requirements. Better infrastructure investment choices, especially from the generation sector which was the main focus of wholesale markets, should offer potential savings and operating practices. This is a directive for evaluating electricity restructuring. If markets do not produce better investment choices than those experienced under the vertically integrated monopoly model, then electricity restructuring will fail. Investors would be expected to do a better job than regulators in balancing location and composition of generation because they would be risking their own money without the safety net that regulators had under rates. Without right incentives for investors, the electricity restructuring would fail.

Defective market design is a problem because it still relies on planners and not markets to keep the demand supplied reliably. However the focus should not be on investors following the directives of central planners. Critical market failures like inadequate scarcity pricing and flaws need to be fixed before the central planner can distance himself from the investor. Limited intervention will be the true liberalization of investors from central planners to maintain the anticipated benefits of electricity restructuring.

The focus of this paper and project is based on an energy-only market. Under this market there is no capacity guarantee put in place to ensure sufficient generation supply. Energy prices fluctuate and when they are high enough, justify new investments. There are many energy-only electricity markets around the world, including the original California market, Nordpool, and the Australian Victoria pool. A shortage of capacity will have the effect of increased prices and increased investment; excess capacity will

drive the market prices down to marginal costs. The price volatility in an energy-only market has high political involvement and has challenged both politicians and regulators. With this market, there is no central resource planning in place to protect resource shortages and make reserves available. Investors will respond only to short-term or spot market price signals. Little investment will take place in low price years causing shortages to develop like what happened in California in the late 1990's. New capacity takes time to build and when there is a lack of planning and coordination, there is generally overbuilding which can lead to very low market prices and deter new investments thus starting the cycle over again. A more price-responsive demand may help moderate these cycles. The energy-only market model will ultimately fail to ensure system reliability and may cause market power concerns.

Overview of Project

Power system reliability, at the transmission level, combined with unit commitment optimal power flow, have been common topics in many of my graduate courses. However, economics and present value analysis studied in this course have opened a new perspective into the past, present, and future infrastructure of the electric grid. My project goal was to combine these three perspectives and to see the effects that individual investors may have on prices, scarcity, reliability, and demand-response. I spent several weeks developing a sophisticated C-based program in MATLAB to simulate investments, competition, load growth, price response, optimal investment strategies, reliability, and profit analysis all based on user direction. The user of course, is you, or anyone who so desires to execute the program. Based on initial settings, many of which can be changed by a simple submenu, the program progresses through a certain period of planning years. During the simulation, the user is given load and price forecasts along with a detailed analysis of investment alternatives. Reliability can be tracked via the contingency analysis option. There will be two main decisions the player (user) can make every year through each time period: invest in new generation, or do not invest in new generation. Competitors, which can be customized in the settings submenu, may invest in new generation decreasing the price forecasts and your profits. The objective for the company players is to, of course, maximize profits. However, if an investment decision is precarious, the player risks losing millions of dollars.

The paper will discuss the simulation process and program functions in detail. High-level flow charts along with pseudo code and tables, will explain the program components. An example simulation will then be presented to illustrate the simulation process. A few different scenarios based on customized settings in the submenus will be presented along with summaries. Applicable uses, enhancements, and other possible program functions will be discussed. Lastly, some concluding remarks will summarize the project work. Appendices contain all applicable code.

II. Simulation Process

Requirements

To run the *Power Sim Investment* program correctly the following are required:

- MATLAB Student Version v13 or greater (developed on v14)
- Optimization Toolbox
- Matpower
- Computer running at least 512 MB RAM (1 GB recommended)

To install the program, create a directory in your MATLAB workspace directory called "*PowerSimInvestor*". Copy and paste the project files to this directory. Open the MATLAB program and set a path to the "*PowerSimInvestor*" folder.

On the command prompt type "mainmenu"
>>mainmenu

If MATLAB fails to recognize this command, you need to review installation and documentation for further assistance.

Overview

Time in the simulation is based on the number of planning years. For example, if the user decides to plan for 5 years, the program will allow five decision years. For each decision year, an optimal power flow is executed and prices determined. Each planning year gives the user two options: invest or continue to the next planning year (do not invest). If the user decides to invest, a generator unit will be added to the system and analyzed like any other generator in the system with respect to least cost optimal dispatch. The investment will take effect at the start of the next planning period. The investor will then be able to see the effective dispatch and annual profits of the generator. A least cost optimal power flow will be in effect for one planning period. Furthermore, the user should also notice that the load in the system increases during the next planning period. *Figure 1* highlights an overview of the time decisions from start to the end of the planning period (N).

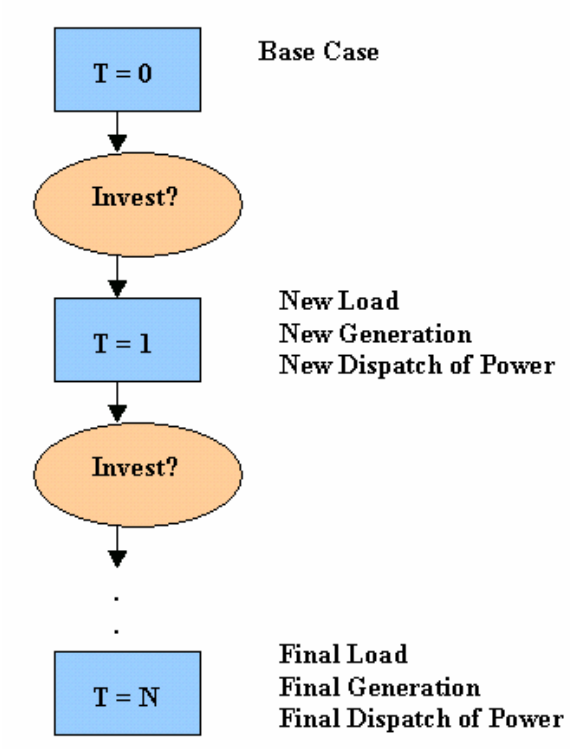
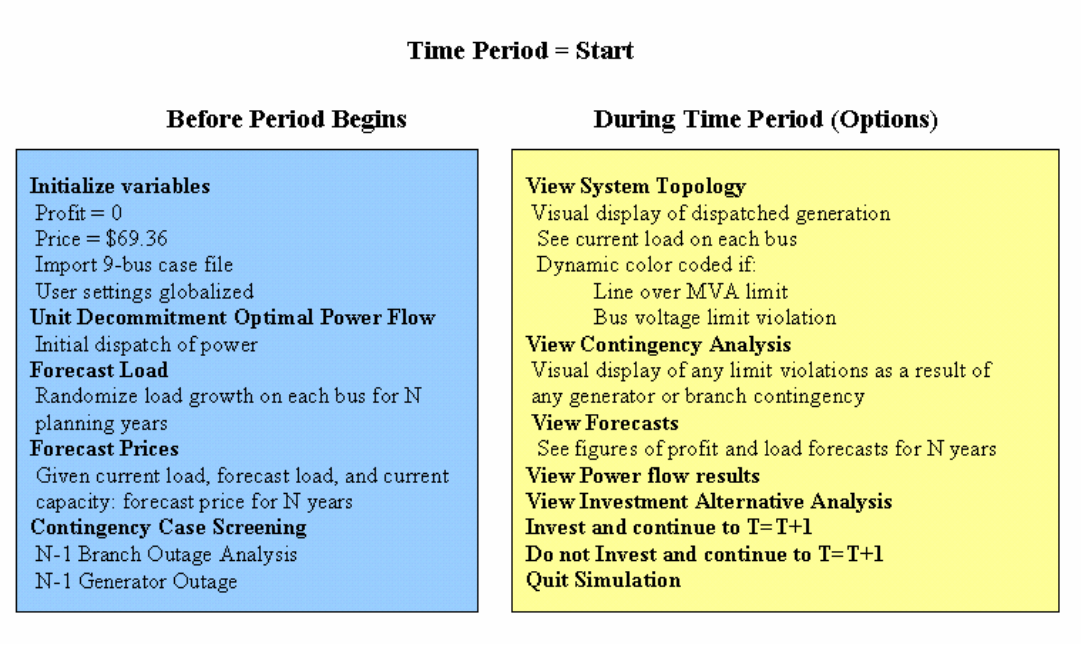


Figure 1 – Time Progression

The first period starts at *t=0* in the year 2006 of the planning period and will plan for the 2006 year. *Figure 2* illustrates a high-level organization of what happens before the user is given a menu of options (*before period begins*) and what options are available (during time period). In order to advance to the next planning year (2007), the user must either invest or choose to not to invest in new generation.

Figure 2, t=0 before and during period.



During the next period (*Figure 3*), the current market price will be the forecast price for this period from the previous period. New generation will be added to the model if the user invested in the last period or if a competitor invested. The forecasted load at each bus for this time period from the previous period will be added to the system model. Another unit decommitment optimal power flow will determine which generators in the

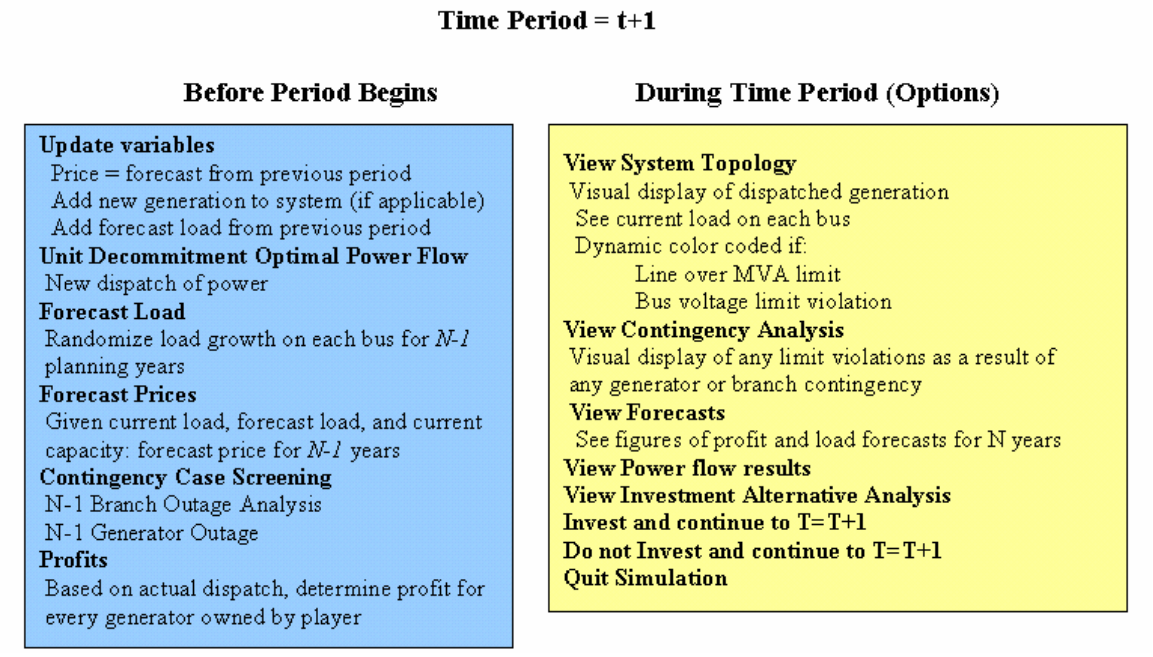


Figure 3, t=t+1 before and during period.

system are committed and at how much generation. Additional forecasts in load and prices will occur but for N-1 periods where N is the number of planning years remaining. Contingency analysis will use the current system model and dispatch as basis to gauge security. Profits for each of the generator will be determined for the current planning year based on the current price and the current year's least cost optimal dispatch. The user will have the same options as the last period and will be able to proceed to the next time simulation

Player Settings

The game officially has two types of players: a coal company investor (default) and a gas company investor. The player settings can only be changed at the beginning of the simulation at the "*Change Settings"* submenu. The player variable only has affect on the cost of the investment and cost to run the plant if it is dispatched. *Table 1* below summarizes the generation unit types for a five-year planning period.

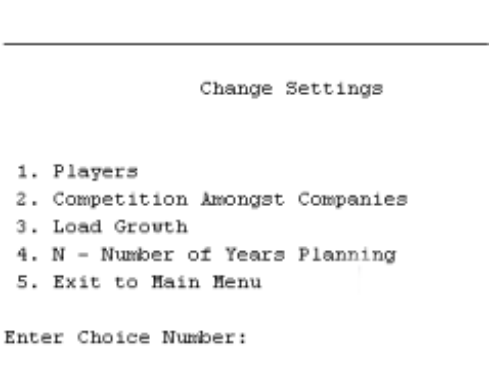


Table 1 – Player Generation Summary for N = 5

Player	MW Capacity	Bidprice	Fixed Levelized Annual Costs $ per MW per YEAR:	Variable Levelized Annual Costs $ per MWH
Coal Company	50	50	268423.7	26.9
Gas Company	51	65	64156	53.54

Generation technologies data sheets can be found in the Appendix.

Competition Settings

Competition makes a simulation more interesting. The competition settings can only be changed at the beginning of the simulation at the "*Change Settings*" submenu. Competitors will be autonomous and invest randomly throughout the simulation, but only after a time period is ended and after a user decided to invest or not invest. *Table 2* shows the probability of an investment activity.

Table 2 – Probability of Competitor Activity

Competition Setting	Probability
OFF	No competitor investment
Low (Default)	25 percent chance there will be a competitor investment
Medium	50 percent chance there will be a competitor investment
High	80 percent chance there will be a competitor investment

If the simulation randomly picks a competitor, the competitor will always invest a 20 MW unit at a bidding price as the same as the current player. For example, if the player is a gas company, the competition will also be a gas company but only investing 20 MW into the system. The investment will then take place at a random bus location without regard to optimal investments. The competition will most likely drive the forecast price down for the next time period if there is excess capacity and possibly reduce the dispatch of the player's generator units invested in the system, and possibly reduce the player's profits. Competitor profits are not tracked during the simulation. The role of the competitor is to add excess capacity in various parts of the system, drive down prices, and influence user's generator investment dispatch and profits.

Load Growth Settings

Load growth settings can be set to either Off (not recommended), Low (Default), Medium, or High. The load growth formula is to take a percentage of the current system load and multiply the growth factor (low = 5%, medium = 10%, and high = 20%) by this number, and then distribute the sum across *each* system bus r*andomly*. A detailed formulation will be discussed in the forecast load process.

Planning Year Settings

The default and recommended number of planning years (and possible decision years) is five. However, the user can enter any number that is reasonable like five or ten. Twenty year planning periods can be done (not recommended); however, the user must be patient for the computer to process, especially when investment analysis is selected.

Simulation Processes

The simulation can be started by choosing option '3' in the main menu. Once a simulation is started, the user cannot change global settings.

```
                    Main Menu


1. How to Play - First time users select this option
2. Change Settings - Choose Player, Select Difficulty
3. Play Simulation
4. Exit
```

A sequence of functions (*refer to Figure 2 – Before Period Begins*) will run and the user should not intervene by any key strokes or mouse clicks until another selection menu appears.

```
Initializing Variables....
Running Base Case and Unit Commitment Optimal Power Flow....
Generating Random Load Growth Profile....
Forecasting Price without Generation Investments........
Running Contingency Screening and Analysis........
N-1 Branch Outage Analysis ........
N-1 Generator Outage Analysis ........
Done Simulating!  ........
```

Initialization of Variables (t=0)

buscoordinates: stores the mapping of the 9-bus system layout so that MATLAB can draw lines on a figure.

profits = 0: when the game begins, the player has not made or lost any money.

intialprice = 69: sets the initial price to $69 per MWH

system default topology: contains initial bus, generator, generator cost, and branch data structures

start year = 2006: the planning period begins at t=0, 2006, regardless of N

MW Capacities: specific to the player, either 50 MW or 51 MW

bidprice: specific to the player.

year matrix: creates an array of years dynamic for N starting with 2006.

Base Case and Unit Commitment Optimal Power Flow

Runs an optimal power flow (Newton's Method) with a heuristic which allows it to shut down "expensive" generators and returns the solved values in data matrices. the objective function value, a flag which is true if the algorithm was successful in finding a solution, and the elapsed time in seconds.

```
options = mpoption('PF_ALG',1,'OPF_ALG',520,'OUT_GEN',1,'OUT_BRANCH',1,'VERBOSE',0);
[baseMVA, bus, gen, branch, success] = runuopf('wscc9bus',options);
```

Unit Decommitment Algorithm (REFERENCE MATPOWER)
MATPOWER includes the capability to run an optimal power flow combined with a unit decommitment for a single time period, which allows it to shut down these expensive units and find a least cost commitment and dispatch. *MATPOWER* uses an algorithm similar to dynamic programming to handle the decommitment. It proceeds through a sequence of stages, where stage *N* has *N* generators shut down, starting with *N* = 0.
The algorithm proceeds as follows:
Step 1: Begin at stage zero (*N* = 0), assuming all generators are on-line with all limits in place.
Step 2: Solve a normal OPF. Save the solution as the current best.
Step 3: Go to the next stage, *N* = *N* + 1. Using the best solution from the previous stage as the base case for this stage, form a candidate list of generators with minimum generation limits binding.
If there are no candidates, skip to step 5.
Step 4: For each generator on the candidate list, solve an OPF to find the total system cost with this generator shut down. Replace the current best solution with this one if it has a lower cost.
If any of the candidate solutions produced an improvement, return to step 3.
Step 5: Return the current best solution as the final solution.

The *wscc9bus* data and structure reference can be found in APPENDIX

Forecasting Load Growth

After the initialization of variables, the forecast load process is executed. This procedure takes an input of the current bus structure (global variable that has a column for load on each bus) and the user defined load growth setting (off, low, medium, or high) and returns a N column, 9 row table with a forecast of load for each bus for N planning years. An example for N=5 planning years is shown in *Figure 4* (below).

Figure 4 – random load forecast for nine-bus system for five years

```
loadforecast =

    0.4511    4.8536    6.9213   15.104    16.121
    2.7613    7.1021   14.225    17.427    25.483
    4.2036     6.737   10.924    17.411    24.178
    2.0061    5.7267   11.701    12.211    19.306
    129.58    134.29    137.73   141.13    148.44
    90.378     90.87    98.709   103.46    109.47
    4.3764    9.9225     10.51   14.654    16.455
    105.37    110.74    115.26   120.62    123.07
    7.3747    10.906    13.287   19.168    24.784
```

Bus #	N=1	N=2	N=3	N=4	N=5
	2006	2007	2008	2009	2010
1	.	.	.	.	.
2	.	.	.	.	.
3	.	.	.	.	.
4	.	.	.	.	.
5	.	.	.	.	.
6	.	.	.	.	.
7	.	.	.	.	.
8	.	.	.	.	.
9	.	.	.	.	.

The algorithm for assigning the load forecast for a bus is simple. Given the current system capacity MW, determine random numbers for each of the 9 busses in the system and sum the random numbers. Next, each bus assigned random number is divided by the sum of the random numbers. The new number will be a "factor" to distribute the load growth amongst each bus. The load forecast for each bus will be the current load on that bus plus the factor times the system MW times the level of growth. The level of growth is either low, medium, or high and changeable only from the main menu.

Figure 5 – Algorithm for Assigning Load Forecast

Level	System Growth
Off	0%
Low	5%
Medium	10%
High	20%

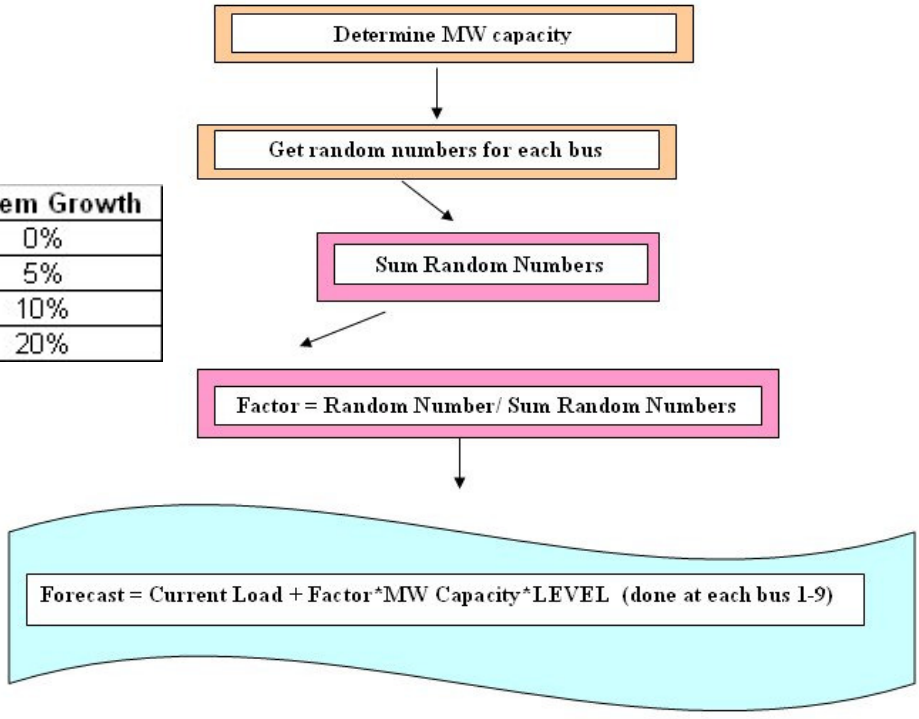


Figure 5 illustrates the algorithm that determines an annual load forecast. *Table 3* shows an example of the current 9-bus system with a total starting load of 315 MW. The current level is set on low with only 5% annual load growth per year. This forecast is input into another run of the algorithm that computes an annual load forecast for another year until the end of the simulation time frame.

Table 3 – Sample Load Forecast for One Year

Current Load	315	Level is Low	0.05
Bus	**Random Number**	**Fraction of Load**	**Load Growth**
1	0.121	0.026	0.413
2	0.451	0.098	1.540
3	0.716	0.155	2.445
4	0.893	0.194	3.050
5	0.273	0.059	0.933
6	0.255	0.055	0.870
7	0.866	0.188	2.957
8	0.232	0.050	0.794
9	0.805	0.175	2.749
total	4.61	1.00	15.75

Figure 6 – Combining Forecast Load to Determine Future Forecast Load for N years

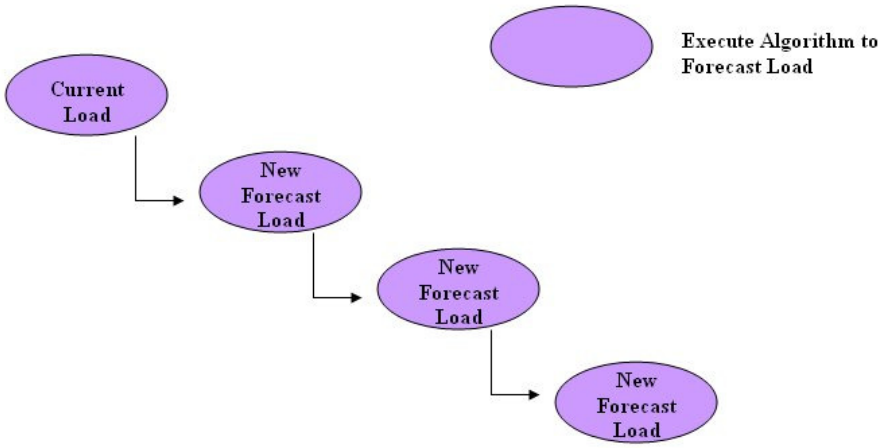


Figure 6 illustrates the recycling of load forecasts for consecutive load forecasts and so on until the end of the simulation period. When the simulation begins, the current load will always be 315 MW. A load forecast for the next year will be feed as the current load into the forecast for following year after that and so on. During the next decision year, the forecast load will be set as the current load. The process will repeat to generate a matrix load forecasts for the remainder of the simulation period.

Figure 7 – View Load Forecast (Under Simulation Menu)

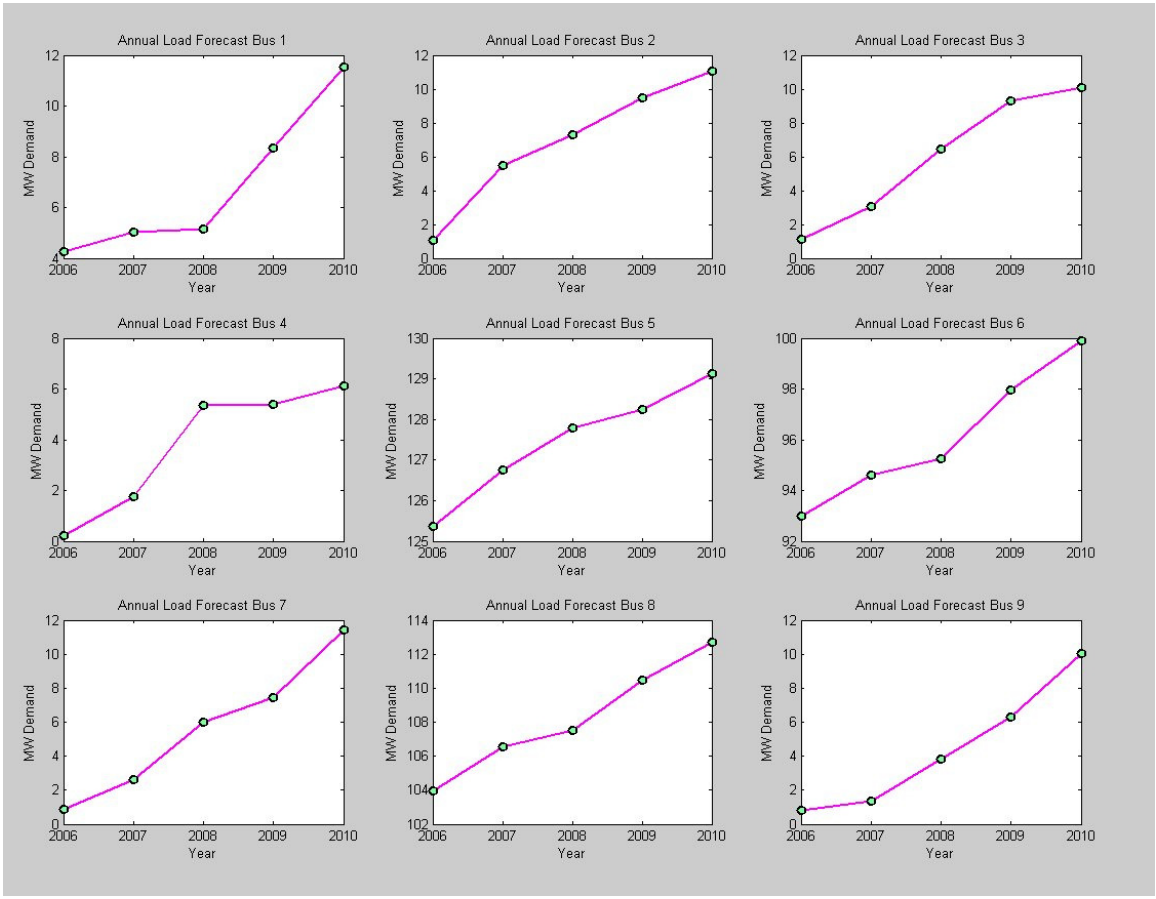


Figure 7 shows the resulting figure of a sample simulation where N= 5 planning years and the load growth level is set to "Low". This figure is accessible for every decision year from the main simulation menu. The load is always increasing from start to finish. The forecast will be randomized for each simulation year. If the user has set the number of planning years to be '5', there should be 5 years of load forecast available during the first planning year. During the second, there will be only '4' available, and so on…

The load forecast routine is the most important process of the project because it influences price, side effects of competitor investments, dispatch of user investments, and profits. All load is considered equal and to be the "demand" of the system. If the generator unit capacity exceeds demand, the load will drive the market price down, and vice versa. Location of load affects only the dispatch of generator units and not price, since the price is based on a total system load and capacity. However, it is important to invest in generation *while* satisfying the demands of the load in order to maximize profits.

Price Forecasts

Another important program component that runs at the beginning of a time period is the price forecaster. This takes the current market price, the current load on the system, the current generation, and determines a price increase or decrease based on capacity vs. load. A simple formula was used to model the price responsiveness of demand based on capacity in the system. Capacity shortages send price signals to investors (the user) to invest. Price caps are not incorporated in this energy-only market. *Prices are forecasted based on lack of investment.* For example, at the beginning of the simulation, if the load growth level is 'high', the price forecast will growth near exponentially. Taking the first few years into consideration should be sufficient when making investment decisions, because more than likely competition will drive down those price forecasts.

Figure 8 – Determination of price based on scarcity

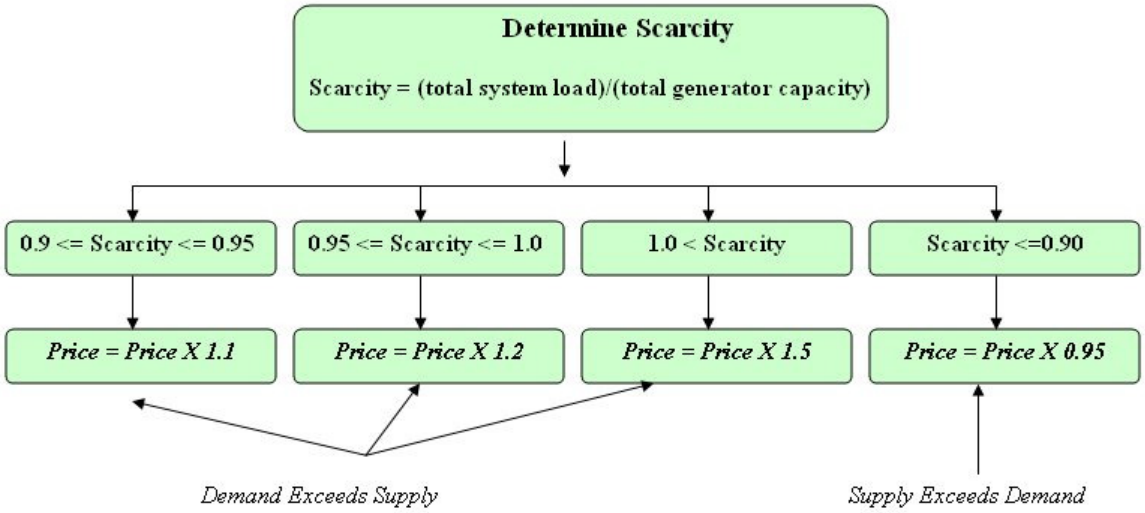
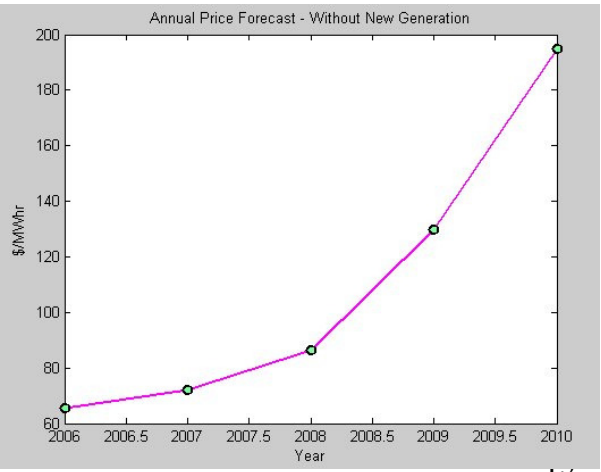


Figure 8 illustrates the calculation of scarcity and the effect on price. For each load forecast, there will be a corresponding price forecast. Sometimes the price will trend downward and at a certain year, trend upward. In a high growth system, the price forecast will almost always trend upward.

Figure 9 – Annual Price Forecast Without New Generation. This is accessible from the main simulation menu for every year of simulation. This shows that there will be a shortage in capacity soon if new investment does not take place.



Contingency Analysis

Contingency analysis was included with this project to help determine the security of the system under new investments and lack of new investments. Originally, an additional player called "Reliability Coordinator", would have been included in the program along with the option to add additional transmission lines to the system. Due to project time constraints, the simulation was simplified to consider only existing transmission lines, buses, and investments by only two types of generation technologies. The contingency analysis was left in the simulation as a tool for the user to study the effects of system security, although the security has no influence on investor profits or anything else for that matter in the simulation. The contingency analysis is stand-alone analysis done every decision year based on *current dispatch* of generation and load. The results will be hidden until the user called the "View Contingency Analysis" option from the main simulation menu. The contingency analysis was developed in another class for a final project (see references) and was slightly modified to analyze a 9-bus system and variable generator outages.

Contingency analysis consists of two main parts: N-1 Generator Analysis, and N-1 Branch outages. N must not be confused with the N variable used to designate the current years of planning. N-1 is a term to describe the security of the system if one component out of a total of N components failed. For example, there are a total of 8 branches in the system. Only 6 of the 8 branches are to be considered for N-1 branch outages because the remaining two would cause an island in the system. N-1 branch outage analysis would remove one branch at a time and analyze the system. If there are any bus voltages outside their normal limits or if there are any branches over their MVA limits, the branch outage and the violations would be recorded. Figure 10 shows a high-level diagram of the contingency analysis

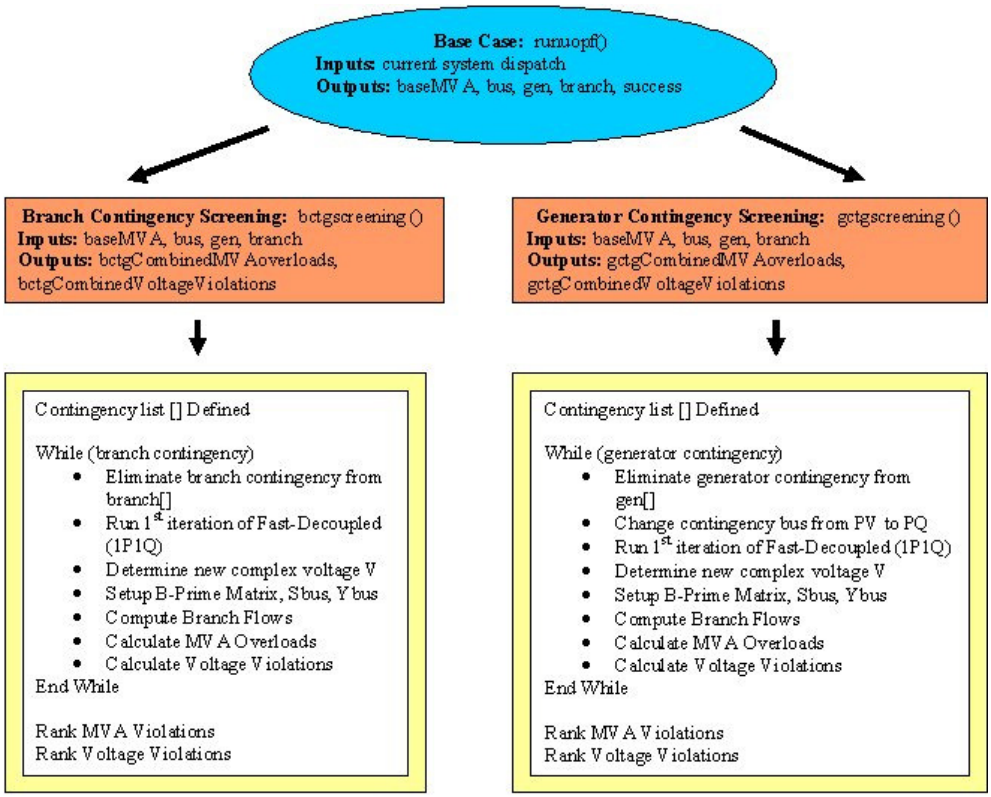


Figure 10 - High-level diagram of contingency analysis program

Branch Contingency Screening:

Description: Screening routine to examine each non-islanding branch outage in the 9-bus system (total of 6 branches). Estimates all post-contingency voltage magnitudes (total of 9 buses) and all post-contingency MVA branch flows (total of 8 branches).

Inputs**:** AC power flow solutions with baseMVA, bus, gen, and branch matrices for the current 9 bus system least cost optimal power flow.

Outputs**:** Condensed matrix with ONLY violated branch MVA overloads and bus voltage violations

bctgCombinedMVAoverloads: (NX5) where N is the number of violations
[<---branch contingency ---> <---branch overloaded--> <---violation--->
[from_bus to_bus from_bus to_bus percent_overload]

bctgCombinedVoltageViolations: (NX9) where N is the number of violations
[<---branch contingency ---> <---bus violated--><----------under voltage---------->
[from_bus to_bus bus current_value low_limit percent_undervoltage

<----------over voltage---------->
current_value high_limit percent_overvoltage]

Sample Results:

bctgCombinedVoltageViolations =

4	5	5	0.87279	0.9	0.96977	0	0	0

The branch contingency screening program examined each non-islanding branch outages in the 9-bus system (total of 6 branches) by considering one contingency at a time, running analysis, and looping to the next contingency while preserving the post-contingency voltage magnitudes (total of 9 buses) and all post-contingency MVA branch flows (total of 8 branches). For each loop iteration, a copy of the solved base case branch matrix is first copied. The branch contingency of interest in the iteration is then removed from the branchcopy matrix.

1P1Q is then run as a fast-decoupled power flow but with only 1 iteration. From the resulting updates in the bus matrix, the voltage magnitudes and angles can be extracted to calculate a complex voltage, V, which is used later in the loop.

A new Sbus and Ybus can now be formed as a result of the updated 1P1Q bus, branch matrices. These Sbus and Ybus matrices along with the new complex voltage, V, are then used to calculate the complex power flows at the from and to end of each branch.

Computing the branch Sf and St (from bus and to bus complex power) now allows comparison against the branch MVA rateA ratings, which were extracted from the 9-bus branch matrix column 6 *(rateA, MVA rating A (long term rating))*. The maximum of the Sf and St was used to compare against the rateA MVA rating of the branch. A percent overload was determined. IF the branch MVA was greater than its rateA limit, the percentage of the overrating was calculated and stored in a MVA violation matrix along with the record or which branch contingency and branch that caused the violation. Non-violations were not saved in the contingency analysis for branches.

The last major function in the program's iterations include the determination of voltage violations. The 1P1Q resultant voltage magnitudes, Vm, were compared against the bus matrix high limit and low limits for voltage magnitudes stored in columns 12 and 13, respectively. A logical comparison determined if the 1P1Q Vm was greater than its high limit and/or less than its low limit. If either of these situations occurred, a percentage of overload severity was calculated for each case. IF there were violations, the results were stored in a violation matrix based on the branch contingency, where the violation occurs, and how severe. Non-violations were not included in the violation matrix. The violation matrix is first organized by high-limit voltage violations, then low limit violations, then sub sorted by their respective severities.

Generator Contingency Screening:

Description: Screening routine to examine each generator unit outage (except the swing) in the 9 bus system (total is dependent upon number of generators in the system). Estimates all post-contingency voltage magnitudes (total of 9 buses) and all post-contingency MVA branch flows (total of 8 branches).

Inputs**:** AC power flow solutions with baseMVA, bus, gen, and branch matrices for the current 9 bus system least cost optimal power flow.

Outputs**:** Condensed matrix with ONLY violated branch MVA overloads and bus voltage violations

gctgCombinedMVAoverloads: (NX4 where N is the number of violations)
[<-generator contingency ---> <---branch overloaded--> <---violation--->
[generator from_bus to_bus percent_overload]

gctgCombinedVoltageViolations: (NX8) where N is the number of violations
[<---generator contingency ---> <---bus violated--><----------under voltage----------->

[generator bus current_value low_limit percent_undervoltage

<----------over voltage---------->
current_value high_limit percent_overvoltage]

The generator contingency screening program examined each generator unit outage (except the swing) in the 9-bus system by considering one contingency at a time, running analysis, and looping to the next contingency while preserving the post-contingency voltage magnitudes (total of 9 buses) and all post-contingency MVA branch flows (total of 8 branches). For each loop iteration, a copy of the solved base case gen matrix is first copied. The generator contingency of interest in the iteration is then removed from the gencopy matrix. In addition, the bus matrix was copied and the bus type of the generator contingency bus was set from PV to a PQ bus.

1P1Q is then run as a fast-decoupled power flow but with only 1 iteration. From the resulting updates in the bus matrix, the voltage magnitudes and angles can be extracted to calculate a complex voltage, V, which is used later in the loop.

A new Sbus and Ybus can now be formed as a result of the updated 1P1Q bus, branch matrices. These Sbus and Ybus matrices along with the new complex voltage, V, are then used to calculate the complex power flows at the from and to end of each branch.

Computing the branch Sf and St (from bus and to bus complex power) now allows comparison against the branch MVA rateA ratings, which were extracted from the 9-bus branch matrix column 6 *(rateA, MVA rating A (long term rating))*. The maximum of the Sf and St was used to compare against the rateA MVA rating of the branch. A percent overload was determined. IF the branch MVA was greater than its rateA limit, the percentage of the overrating was calculated and stored in a MVA violation matrix along with the record or which generator contingency and branch that caused the violation. Non-violations were not saved in the contingency analysis for generators

The last major function in the program's iterations include the determination of voltage violations. The 1P1Q resultant voltage magnitudes, Vm, were compared against the bus matrix high limit and low limits for voltage magnitudes stored in columns 12 and 13, respectively. A logical comparison determined if the 1P1Q Vm was greater than its high limit and/or less than its low limit. If either of these situations occurred, a percentage of overload severity was calculated for each case. IF there were violations, the results were stored in a violation matrix based on the generator contingency, where the violation occurs, and how severe. Non-violations were not included in the violation matrix. The violation matrix is first organized by high-limit voltage violations, then low limit violations, then sub sorted by their respective severities.

Determination of Profits

$PROFIT_{GENERATOR}$ =

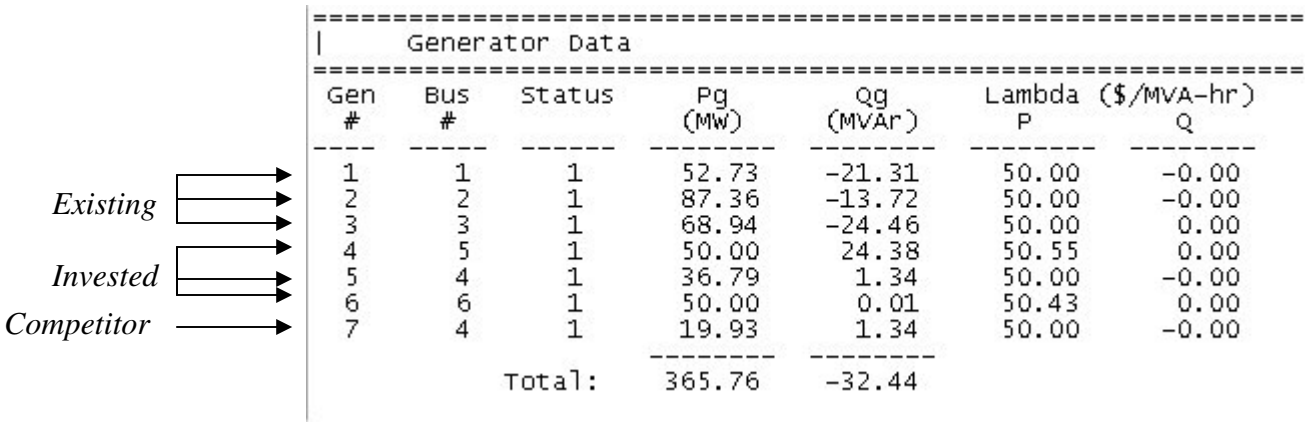
$[MW_{DISPATCHED}$ X **P** X 8760] – [**A** X $MW_{CAPACITY}$ + **B** X $MW_{DISPATCHED}$ X 8760] /10^6

Where:

A = Fixed Levelized Annual Costs $ per MW per YEAR
B = Variable Levelized Annual Costs $ per MWH
P = Current Market Price

For each generator the investment player owns, an annual profit is computed based on the actual dispatch of generator for that unit, the unit's capacity, and the current market price $/MWH . Profits are computed only if there is a generator in the system that the user invested. For example, if the user decides to invest a generator, the profits will not be determined until the next time period. If the user invested more than one generator, profits will be determined individually and then summed. Negative profits are possible if the unit decommitment power flow analysis did not commit a user's generator, or if the generator had a MW dispatch that did not cover the costs. The simulation only shows a combined net profit for all generators by the user and not individual unit profits. The user can run the "View Base Case and Unit Commitment Optimal Power Flow" from the simulation main menu and see which generators he invested are committed and at how many MW. Part III (Simulation Menu) will illustrate and detail the results from this menu option.

Figure 11 – Example of Generators on the System and Current Dispatch



Gen #	Bus #	Status	Pg (MW)	Qg (MVAr)	Lambda ($/MVA-hr) P	Lambda ($/MVA-hr) Q
1	1	1	52.73	-21.31	50.00	-0.00
2	2	1	87.36	-13.72	50.00	-0.00
3	3	1	68.94	-24.46	50.00	0.00
4	5	1	50.00	24.38	50.55	0.00
5	4	1	36.79	1.34	50.00	-0.00
6	6	1	50.00	0.01	50.43	0.00
7	4	1	19.93	1.34	50.00	-0.00
		Total:	365.76	-32.44		

Figure 11 illustrates an example of a Coal Company player investing a coal unit generator on bus 4,5, and 6. Two of the investments resulted in full dispatch of his unit, which means maximum profits. However, the investment on bus 4 did not result in optimal dispatch of his unit.

III. Simulation Menu

Figure 12 – Simulation Main Menu

```
                Simulation

   Player:  Coal Company
                            Total Profits: 0
                                     Year: 2006


 1.  View Current System Topology (Figure)
 2.  View Current Load Growth Forecast (Figure)
 3.  View Current Annual Price Forecast (Figure)
 4.  View Current System Topology with N-1 Contingency Analysis (Figure)
 5.  View Base Case and Unit Commitment Optimal Power Flow
 6.  View Investment Alternatives Analysis
 7.  Invest Generation and Move to Next Year
 8.  Do Not Invest Generation and Move to Next Year
 9.  Quit Simulation and Exit to Main Menu

Enter Choice Number:
```

The simulation main menu consists of nine options as shown above in *Figure 12.* The player will be shown as well as the current total profits and the current planning year. The main simulation menu will be the same throughout the entire simulation. Profits will be updated for each year.

View Current System Topology (Figure)

The current system topology can be accessed from typing '1' in the main simulation menu. A figure in MATLAB will appear minimized. Bring the MATLAB figure to view. *Figure 13* shows the system topology at the beginning of every simulation. Each bus in the system (total of 9) is drawn as a thick white line over a black background. The transmission lines are drawn as thin white lines. Bus voltages (in per unit) are displayed above their respective bus bars. The bus names are on the left-hand side of the bus bar. Generation is shown has yellow text with both MW and MVAR. Load is also shown but colored as magenta. The current system topology will dynamically color code the transmission lines to RED if they exceed their MVA ratings. The bus voltages will also color code to RED if they exceed their high voltage limit. The bus voltages will color BLUE if they are less than the low voltage limit. *Figure 14* shows another display of the system topology with additional invested generation and load growth. The display will show current load and dispatched generation for the planning year.

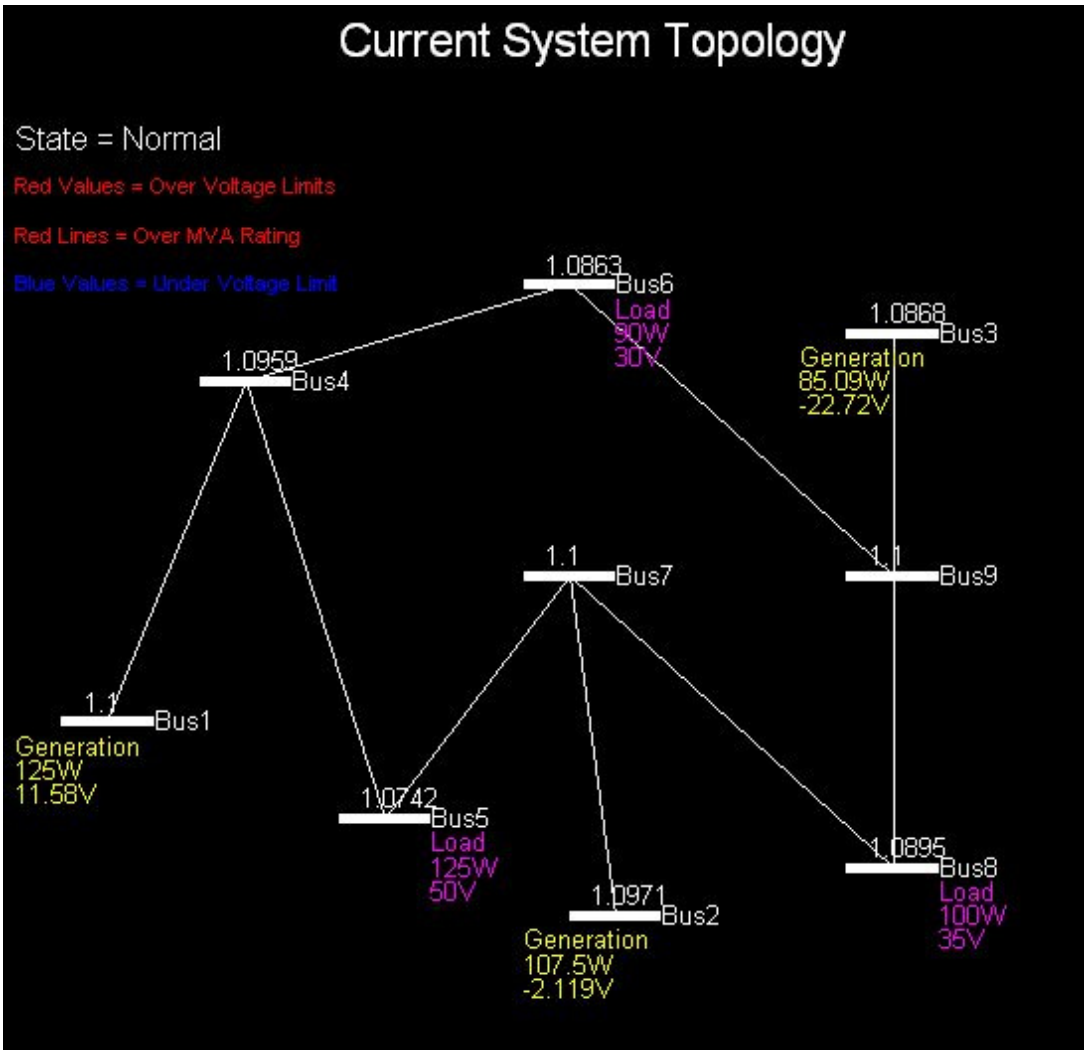


Figure 13 – Starting Current System Topology Figure

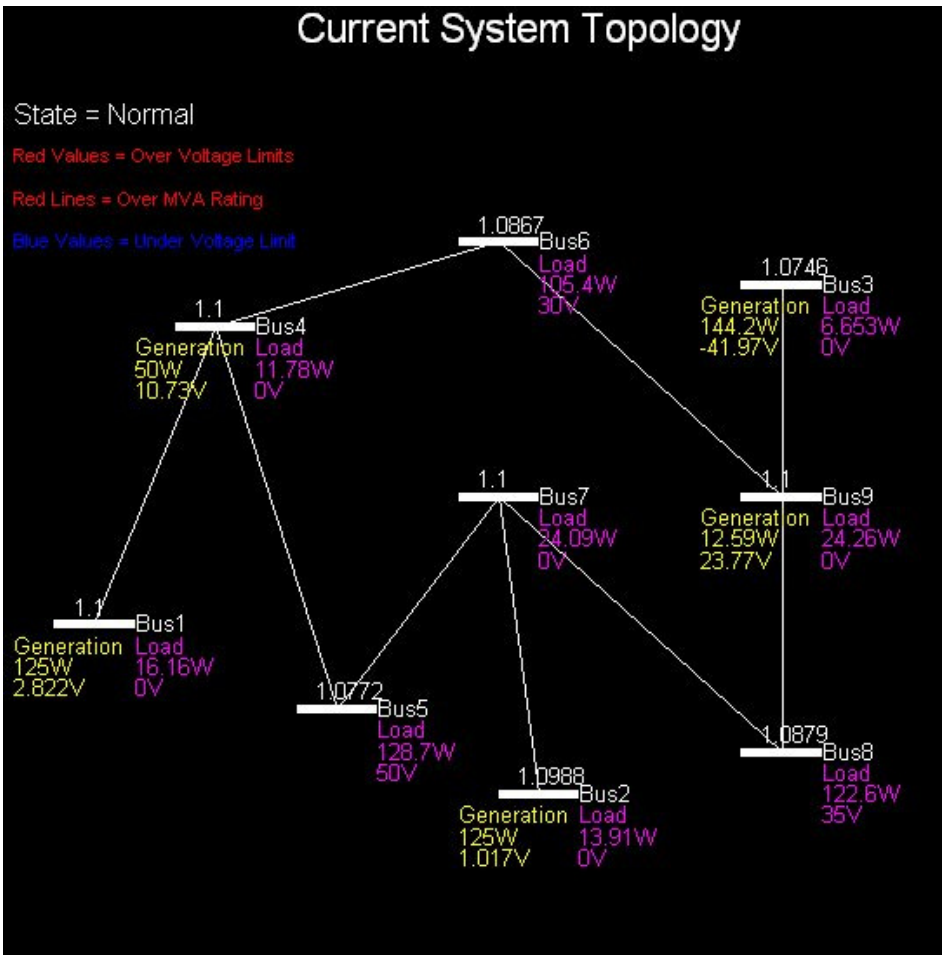
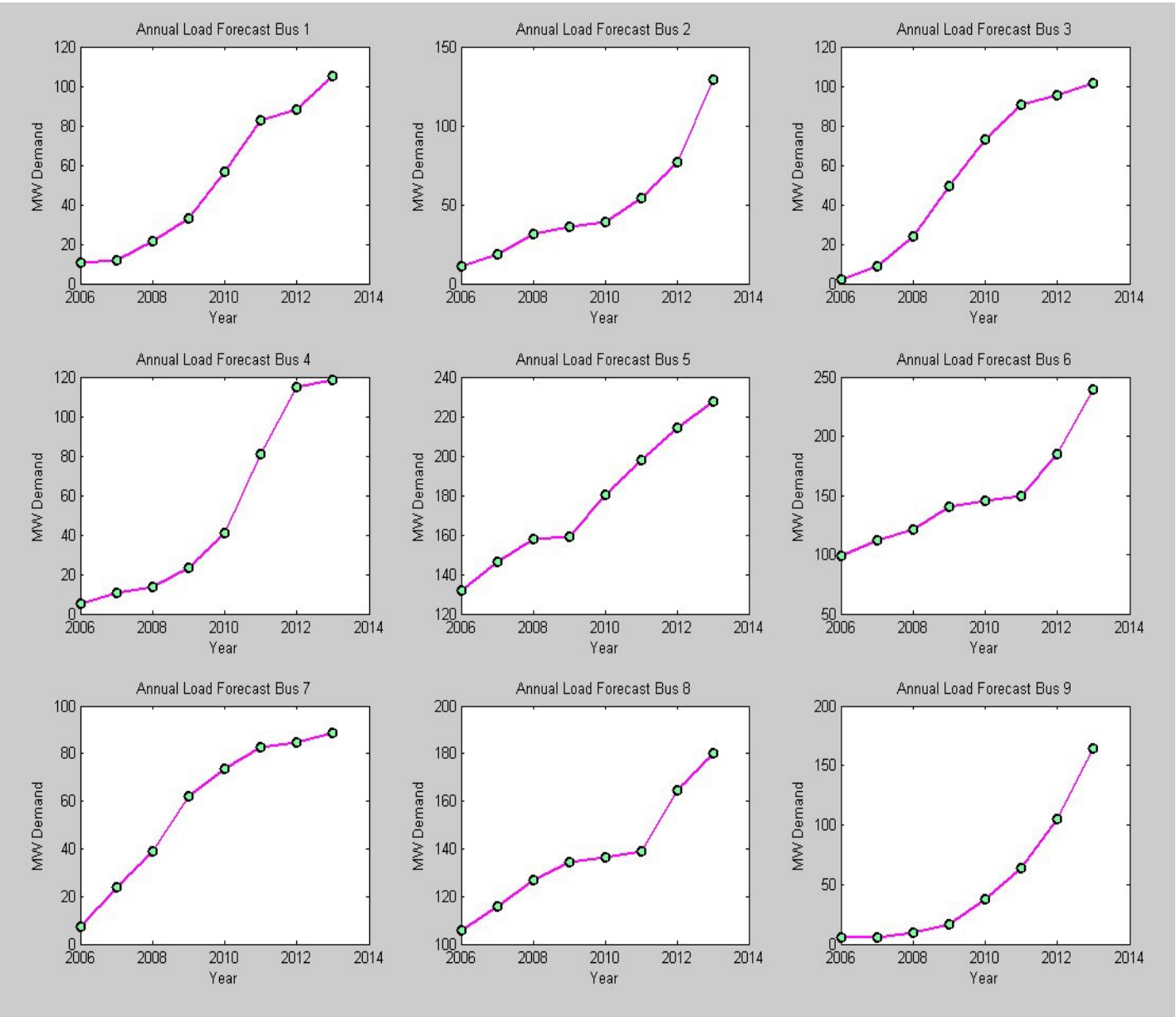


Figure 14 – System Topology with Load Growth and Additional Generators

View Current Load Growth Forecast (Figure)

The load growth forecast figure can be accessed from typing '2' in the main simulation menu. A figure in MATLAB will appear minimized. Bring the MATLAB figure to view. *Figure 15* shows the individual bus load forecasts for years current through the remainder of the simulation. The load forecast process was explained under the simulation process. This menu option subplots the individual bus forecasts based on the load forecast from the beginning of the simulation year. The load forecast display will change every simulation year because it is random. The load forecast for the next planning will be added to the system.

Figure 15 – Load Forecasts on Each Bus



View Current Annual Price Forecast (Figure)

The price forecast figure can be accessed from typing '3' in the main simulation menu. A figure in MATLAB will appear minimized. Bring the MATLAB figure to view. *Figure 16* shows the price forecasts for years current through the remainder of the simulation. The price forecast process was explained under the simulation process. This menu option plots the forecast prices for the current year through the end of the simulation. The price forecast is based on no new generation in the system and should entice the investor to invest in new generation. Competition in the system can bring the forecast down to reasonable prices.

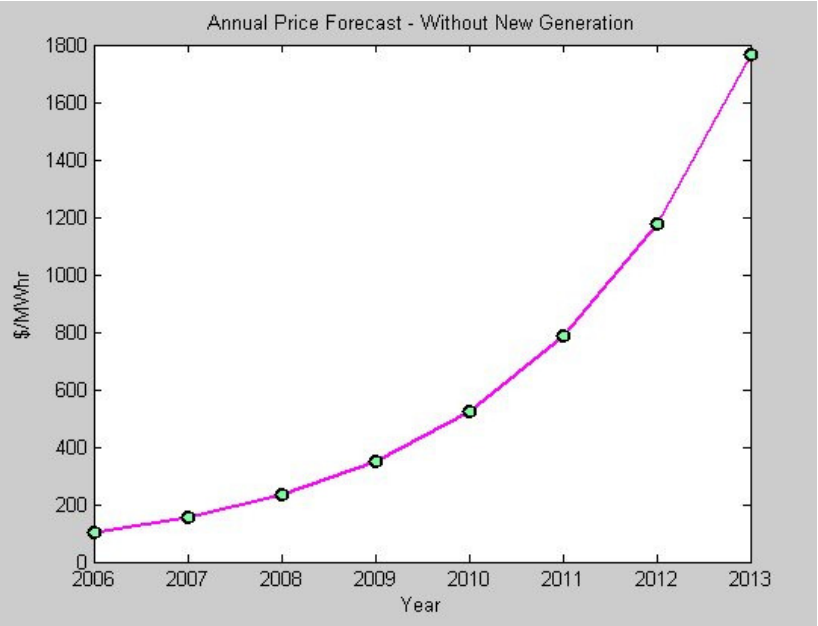
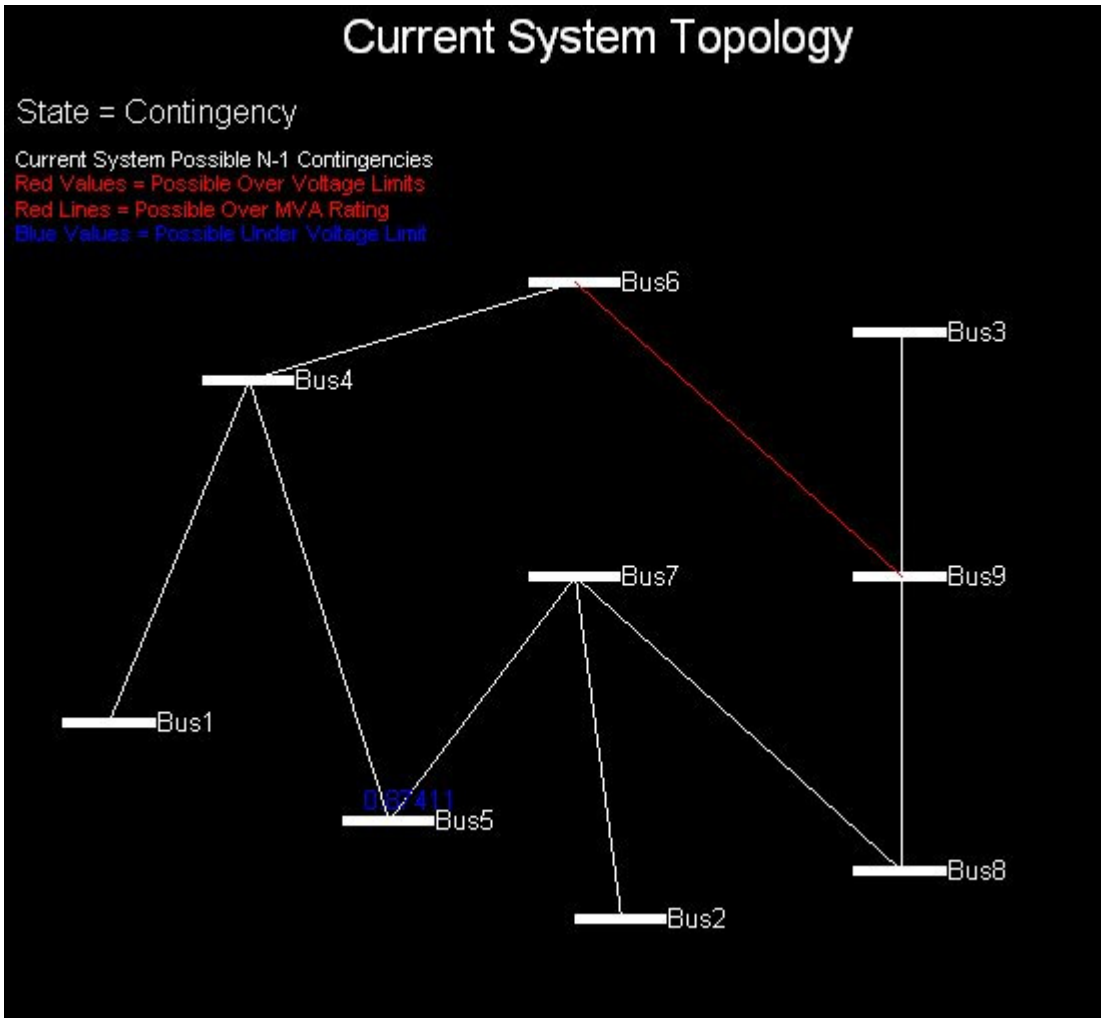


Figure 16 – Annual Price Forecast – Without New Generation

View Current System Topology with N-1 Contingency Analysis (Figure)

The current system topology with N-1 contingency analysis can be accessed from typing '4' in the main simulation menu. A figure in MATLAB will appear minimized. Bring the MATLAB figure to view. *Figure 16* shows an example of current system that is vulnerable to a line(6-9) being overloaded and bus 5 being under voltage . Each bus in the system (total of 9) is drawn as a thick white line over a black background. The transmission lines are drawn as thin white lines. Bus voltages (in per unit) are displayed above their respective bus bars IF there is a limit violation. The bus names are on the left-hand side of the bus bar. Generation and load are not shown on this figure. The current system topology will dynamically color code the transmission lines to RED if they exceed their MVA ratings from ANY N-1 contingency. The bus voltages will also color code to RED if they exceed their high voltage limit from ANY N-1 contingency. The bus voltages will color BLUE if they are less than the low voltage limit from ANY N-1 contingency. The results of the contingency analysis done at the beginning of the simulation year will also display in the MATLAB window when the menu option is called. The contingency analysis was detailed in the simulation process.

Figure 16 – Contingency Analysis Shows Possible Line(6-9) Overload and Bus5 under voltage alarms.



N-1 Branch Contingency Results

Screening routine to examine each non-islanding branch outage in the 9 bus system.
Estimates all post-contingency voltage magnitudes and all
post-contingency MVA branch flows.

Inputs: Base case AC power flow solutions with baseMVA, bus, gen, and
branch matrices for the 9 bus system
Outputs: Condensed matrix with ONLY violated branch MVA overloads and bus voltage
violations

MVA Overloads Due to Any Branch Outage

bctgCombinedMVAoverloads: (NX5) where N is the number of violations
[<---branch contingency ---> <---branch overloaded--> <---violation--->
[from_bus to_bus from_bus to_bus percent_overload]

bctgCombinedMVAoverloads =

5 7 6 9 125.3

Voltage Violations Due to Any Branch Outage

bctgCombinedVoltageViolations: (NX9) where N is the number of violations

[<---branch contingency ---> <---bus violated--><----------under voltage----------><--------over voltage----------
-->
[from_bus to_bus bus current_value low_limit percent_undervoltage current_value high_limit
percent_overvoltage]

bctgCombinedVoltageViolations =

4 5 5 0.87411 0.9 0.97123 0 0 0

N-1 Generator Contingency Results

Screening routine to examine each generator unit outage (except the swing) in the
9 bus system. Estimates all post-contingency voltage magnitudes
and all post-contingency MVA branch flows.

Inputs: Base case AC power flow solutions with baseMVA, bus, gen, and
branch matrices for the 9 bus system
Outputs: Condensed matrix with ONLY violated branch MVA overloads and bus voltage
violations

MVA Overloads Due to Any Generator Outage

gctgCombinedMVAoverloads: (NX4 where N is the number of violations
[<-generator contingency ---> <---branch overloaded--> <---violation--->
[generator bus from_bus to_bus percent_overload]

gctgCombinedMVAoverloads =

Empty matrix: 0-by-4

Voltage Violations Due to Any Generator Outage

gctgCombinedVoltageViolations: (NX8) where N is the number of violations

[<---generator contingency ---> <---bus violated--><----------under voltage----------><--------over voltage----------
----->
[generator bus bus current_value low_limit percent_undervoltage current_value high_limit
percent_overvoltage]

gctgCombinedVoltageViolations =

Empty matrix: 0-by-8

View Base Case and Unit Commitment Optimal Power Flow

Selecting option '5' from the main menu will run the unit decommitment optimal power flow and return the results (the same results done at the beginning of the time period). Important sections to consider is the system summary and the generator data. Under "System Summary" the number or generators and committed generators are shown, along with the total generator capacity, actual generation, load, and losses. The generator data lists each generator, the bus it is on, the status, and the amount of MW on-line. A sample optimal power flow is shown below:

```
Converged in 0.39 seconds
Objective Function Value = 22739.38 $/hr
==========================================================================
|     System Summary                                                    |
==========================================================================

How many?                How much?              P (MW)            Q (MVAr)
---------------------    --------------------  -------------  -----------------
Buses               9    Total Gen Capacity     570.0         -360.0 to 450.0
Generators          6    On-line Capacity       570.0         -360.0 to 450.0
Committed Gens      6    Generation (actual)    454.8              -25.6
Loads               9    Load                   453.6              115.0
  Fixed             9      Fixed                453.6              115.0
  Dispatchable      0      Dispatchable           0.0 of 0.0         0.0
Shunts              0    Shunt (inj)            0.0                0.0
Branches            9    Losses (I^2 * Z)       1.19              22.80
Transformers        0    Branch Charging (inj)   -                 163.4
Inter-ties          0    Total Inter-tie Flow   0.0                0.0
Areas               1

                          Minimum                      Maximum
                  ------------------------  -------------------------------
Voltage Magnitude   1.082 p.u. @ bus 3          1.100 p.u. @ bus 4
Voltage Angle      -5.66 deg   @ bus 8          0.00 deg   @ bus 1
P Losses (I^2*R)             -                 0.37 MW    @ line 4-5
Q Losses (I^2*X)             -                 5.44 MVAr  @ line 2-7
Lambda P           50.00 $/MWh @ bus 7         50.59 $/MWh @ bus 5
Lambda Q           -0.00 $/MWh @ bus 6          0.00 $/MWh @ bus 1

==========================================================================
|     Generator Data                                                     |
==========================================================================
 Gen   Bus   Status     Pg        Qg       Lambda ($/MVA-hr)
  #     #               (MW)     (MVAr)      P         Q
----  -----  ------  --------  --------  --------  --------
  1      1       1    127.98    -14.51     50.00      0.00
  2      2       1    118.85    -22.45     50.00      0.00
  3      3       1     87.96    -30.98     50.00      0.00
  4      5       1     50.00     24.92     50.59     -0.00
  5      6       1     50.00      0.05     50.44     -0.00
  6      8       1     20.00     17.40     50.44      0.00
                     --------  --------
            Total:    454.79    -25.57

==========================================================================
|     Bus Data                                                           |
==========================================================================
 Bus      Voltage          Generation             Load          Lambda($/MVA-hr)
  #   Mag(pu) Ang(deg)   P (MW)   Q (MVAr)   P (MW)   Q (MVAr)     P        Q
-----  ------- --------  --------  --------  --------  --------  -------  -------
    1  1.091    0.000    127.98    -14.51     25.00      0.00    50.000     -
    2  1.086   -0.707    118.85    -22.45     20.06      0.00    50.000     -
    3  1.082   -1.265     87.96    -30.98     13.50      0.00    50.000     -
    4  1.100   -2.833       -         -       2.63      0.00    50.000     -
    5  1.095   -5.530     50.00     24.92    140.35     50.00    50.589     -
```

```
    6  1.095   -4.306     50.00      0.05     95.12     30.00    50.443     -
    7  1.100   -3.670       -          -      16.53      0.00    50.000     -
    8  1.096   -5.659     20.00     17.40    125.49     35.00    50.438     -
    9  1.100   -3.365       -          -      14.92      0.00    50.000     -
                        --------  --------  --------  --------
               Total:    454.79    -25.57    453.60    115.00

================================================================================
|     Branch Data                                                              |
================================================================================
Brnch   From   To    From Bus Injection   To Bus Injection     Loss (I^2 * Z)
  #     Bus    Bus    P (MW)   Q (MVAr)   P (MW)   Q (MVAr)   P (MW)   Q (MVAr)
-----  -----  -----  --------  --------  --------  --------  --------  --------
   1      1      4    102.97    -14.51   -102.97     19.74     0.000      5.23
   2      2      7     98.79    -22.45    -98.79     27.89     0.000      5.44
   3      3      9     74.46    -30.98    -74.46     34.23     0.000      3.25
   4      4      5     66.74     -9.86    -66.37     -8.20     0.368      3.13
   5      4      6     33.61     -9.88    -33.45     -8.30     0.159      0.86
   6      5      7    -23.98    -16.88     24.13    -19.19     0.154      0.78
   7      6      9    -11.67    -21.65     11.72    -21.30     0.044      0.19
   8      7      8     58.13     -8.71    -57.89     -7.24     0.237      2.01
   9      8      9    -47.60    -10.36     47.82    -12.93     0.225      1.91
                                                             --------  --------
                                                   Total:     1.188     22.80

================================================================================
|     Voltage Constraints                                                      |
================================================================================
Bus #  Vmin mu    Vmin     |V|   Vmax    Vmax mu
-----  --------   -----   -----  -----   --------
    4       -     0.900   1.100  1.100    43.573
    7       -     0.900   1.100  1.100    41.771
    9       -     0.900   1.100  1.100    23.712

================================================================================
|     Generation Constraints                                                   |
================================================================================
 Gen   Bus                Active Power Limits
  #     #    Pmin mu    Pmin       Pg       Pmax     Pmax mu
----  -----  -------  --------  --------  --------  -------
   4     5       -      10.00     50.00     50.00    0.589
   5     6       -      10.00     50.00     50.00    0.443
   6     8       -      10.00     20.00     20.00    0.438
```

View Investment Alternatives Analysis

Selection of option ‘6’ from the simulation main menu will execute the investment alternative analysis and return results. Investment Analysis is done on the user’s request because of the considerable time it takes to complete. Depending on the time period being analyzed, the size of the system, the analysis can take anywhere between 10 seconds to 5 minutes to complete. The analyze investment analysis process is very complex. It simulates adding a generator unit at a particular bus. Optimal power flow is run for that investment for the current year and every year to the end of the simulation period. The current load forecast is used in the yearly power flow calculations. Based on the load forecast, a calculation of how much MW from the investment alternative is committed from optimal power flow. This process is repeated for 9 bus alternatives until a “Unit Dispatch Forecast” (shown in results below) is completed. Price forecasts are used with the “Unit Dispatch Forecast” to compute the predicted revenues for each alternative. Profits are revenues minus the marginal costs (see Unit Dispatch

Profit Forecast under results). Finally a "Net Present Value Analysis" is calculated based on the "Unit Dispatch Profit Forecast" profits discounted to the current year at a rate of 10%. Results of a sample analysis for an investor are shown below. *Figures 17 and 18* illustrate the algorithms involved in forecasting the MW dispatched and profits for each possible investment alternative done at the current year. There is only one type of investment analysis presented in the simulation: Investment at the current time period (no delayed investments).

Unit Dispatch Forecast

Based on your bidding price of: 75
Optimal Power Flow with Unit Decommitment Optimization was executed for the planning period. Based on a forecasted load growth at each bus during the planning period, your bid, and your unit capacity, the following table shows a forecast of how your investment would be dispatched IF you invested a unit at the bus location at time t=0.

Invested at Bus	Year1	Year2 ....
Bus1	P1	P2 ....
Bus2	P1	P2 ....
.	.	. ....
.	.	. ....

P = active power in MW dispatched from YOUR invested unit

bus_forecast =

Columns 1 through 10

0	2006	2007	2008	2009	2010	2011	2012	2013	2014
1	0	10	62.617	79.38	103.24	123.5	129.41	203.2	171.12
2	0	10	61.713	88.697	103.89	106.62	130.58	152.95	193.17
3	0	10	60.767	79.913	112.63	139.48	130.55	185.71	228.29
4	0	10	62.016	79.378	94.84	119.5	147.66	241.78	207.58
5	0	10	62.458	78.931	94.673	119.42	151.14	184.45	207.88
6	0	10	60.425	79.237	94.622	102.02	151.08	145.53	207.68
7	0	10	62.261	79.906	95.387	139.67	130.58	172.44	193.23
8	0	10	61.443	79.75	95.254	119.53	130.45	159.12	193.38
9	0	10	61.737	79.867	112.66	139.46	177.59	172.47	193.45

Column 11

2015
208.87
217.85
234.61
252.45
252.95
255.26
217.85
234.31
217.88

Comments: *Adding a 50 MW gas turbine into the system with the current load forecasts would not result in a dispatch until 2007 due to excess generation in the system. However, if competitors do not invest, it will be economical to invest in a gas turbine generator during the 2006 planning year for 2007.*

Unit Dispatch Profift Forecast

Based on your bidding price of: 75
AND on your unit annual marginal cost of: 64364 63
Considering the unit dispatch forecast above with the market price forecast, an investment analysis table is shown below.

	Year1	Year2 ....
Market Price (Forecast)	$/MW	$/MW ....
Profit at Bus1 Investment	$ M	$ M ....

```
 Profit at Bus2 Investment        $ M      $ M    ....
    .                      .      .     ....
    .                      .      .     ....
```

```
profit_forecast =

 Columns 1 through 10

         0      2006      2007      2008      2009      2010      2011      2012      2013      2014
         0     65.55    98.325    147.49    221.23    331.85    497.77    746.66      1120      1680
         1         0    2.4368    42.225    104.81    236.36    462.22    766.53    1868.1    2412.6
         2         0    2.4368    41.616    117.11    237.84    399.05    773.41    1406.1    2723.5
         3         0    2.4368    40.978    105.51    257.84    522.05    773.28    1707.3    3218.7
         4         0    2.4368    41.821    104.81    217.12    447.25    874.59    2222.7    2926.7
         5         0    2.4368    42.118    104.22    216.74    446.95    895.22    1695.7    2930.9
         6         0    2.4368    40.747    104.62    216.62    381.84    894.85    1337.9      2928
         7         0    2.4368    41.986     105.5    218.37    522.77    773.43    1585.3    2724.3
         8         0    2.4368    41.434     105.3    218.07    447.37    772.64    1462.9    2726.4
         9         0    2.4368    41.632    105.45    257.92    521.96    1051.9    1585.6    2727.4

 Column 11

      2015
      2520
    4481.7
    4674.4
    5034.1
    5416.8
    5427.6
    5477.2
    4674.5
    5027.6
      4675
```

```
 Net Present Value Analysis


 Based on the above forecasts and estimates, a net present value table
 of future profits discounted, r = 0.1, to present value is shown below.
 You should invest in adding generation at the bus with the highest
 "expected" value (Net Present Value) of money


         $ Profits for Entire Planning Period as NPV
 Bus1                  $ M Dollars
 Bus2                  $ M Dollars
  .                    .
  .                    .

  **** Remember which bus that you would like to invest in (1-9) at this time
```

```
npv =

         1    4528.9
         2    4496.3
         3    5058.8
         4    5307.6
         5    5077.7
         6    4890.9
         7    4630.3
         8    4666.6
         9    4798.7
```

Comments: *Net present value analysis should not be the only profit maximizing number considered, although it is a good indictor of where to invest. High competition in the system can drastically reduce the forecasts after a few years into the simulation.*

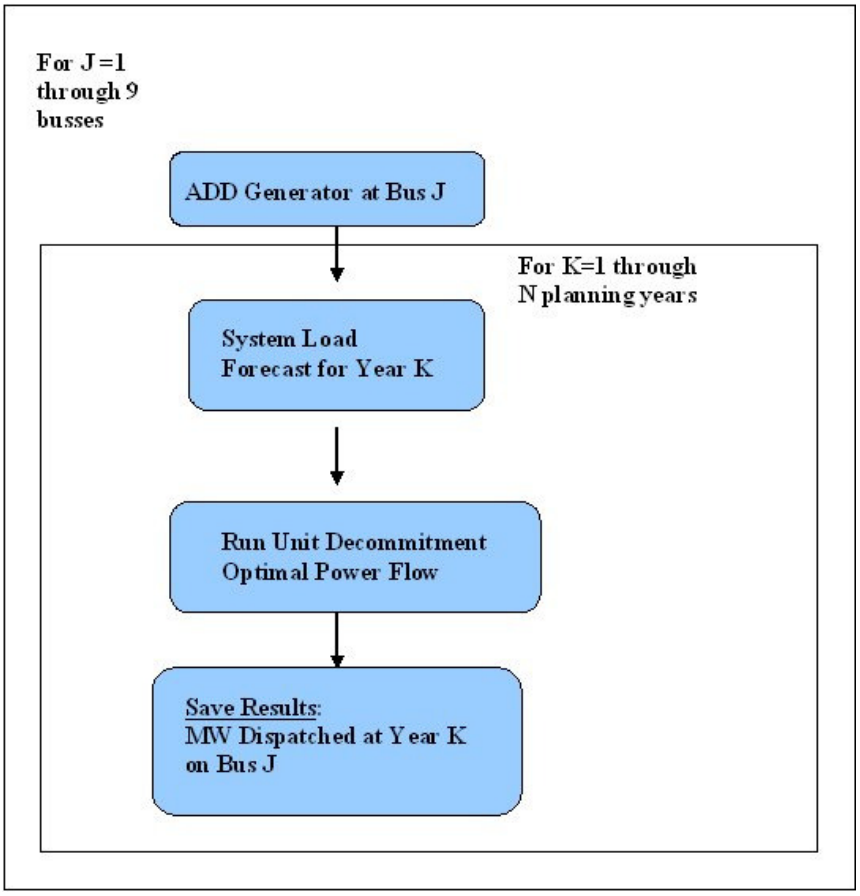
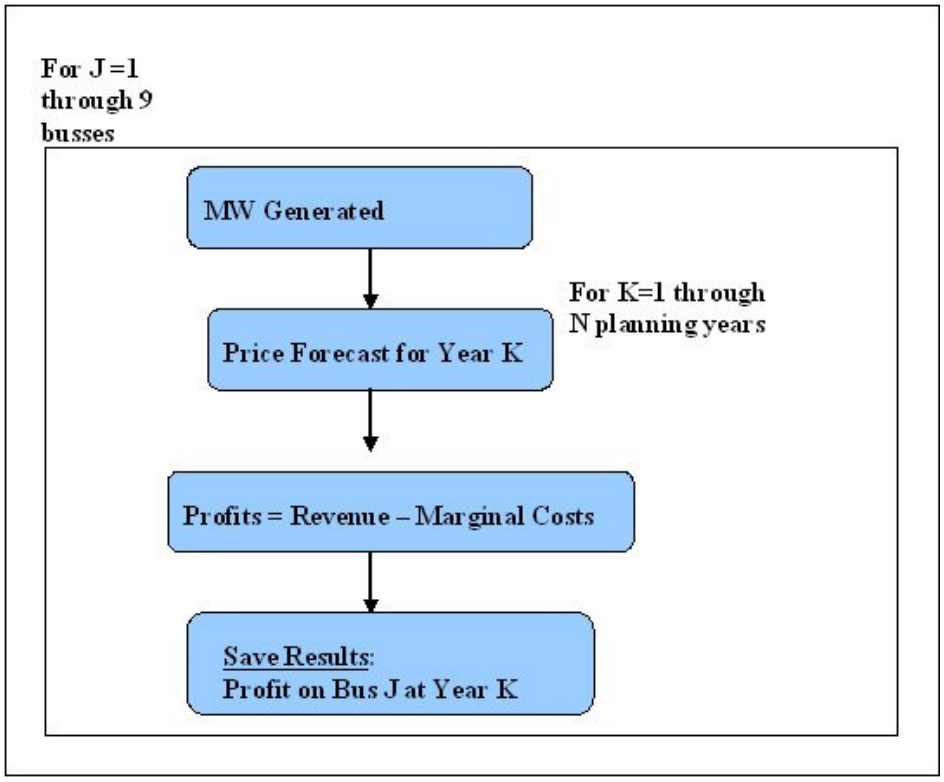
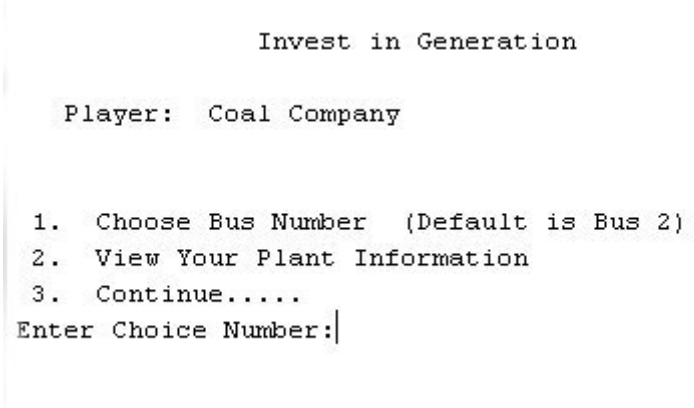


Figure 17 – Overview of the investment alternative analysis algorithm for 9 system busses during a simulation period of N years forecast dispatch of generation

Figure 18 – Overview of the investment alternative analysis algorithm for 9 system busses during a simulation period of N years forecast of yearly profits



Invest Generation and Move to Next Year



The user has the option to invest in new generation for the next year by choosing option '7' from the simulation main menu. A submenu "*Invest in Generation*" will appear. The user has the option of choosing which bus to invest by selecting choice '1', and viewing the generator plant characteristics by option choice '2'. After the user has selected a bus to invest 50 MW of new generation (either coal or gas depending on the player), select '3' to continue and advance to the next planning year. There is a chance that a competitor will also invest after the user has made his investment decision. This notice will appear momentarily after the user selects 'Continue'. The simulation will then advance to the next time period.

Do Not Invest Generation and Move to Next Year

The user has the option to forgo investments and just advance to the next year in the simulation by selecting option '8' from the simulation main menu. There is a chance that a competitor will also invest after the user has selected this option. This notice will appear momentarily after the user selects the option to not invest. The simulation will then advance to the next time period.

Quit Simulation

Selecting option '9' from the simulation main menu at any time period will revert the program back to initial main menu where the global settings (player, competition, load growth, number years of planning) will remain saved from the previous simulation. To exit the program entirely from the main menu, select option '4' to exit the program.

IV. Scenarios and Results

Scenarios

Setting the player, load growth, competition, and N planning year settings before the simulation will give interesting results depending on the settings. There are two types of players, 4 load growth settings, 4 competition level settings, and N planning year possibilities. The user may choose to simulate low load growth with high competition, which will more than likely deplete the market price to the marginal costs. Another scenario would be to simulate a high load growth and low competition, which can result in windfall profits for the investor. An important fact about competitors is that if the market price is below their bidding price (which is the same as yours), they will not invest in the system! Some scenarios are outlines below:

Player = Coal Company, Competition = High, Load Growth = Low

The simulation starts with a high expectation of net present value profit if the player were to invest at the current time period. Further inspection of the individual profit years shows that investing in the system at the current year would drive down the market prices and lead to negative profits in a few years until the load growth demand brings the market price towards profitability. Since competition is high, the competitors will drive down market prices even further. Investing in the first year will result in positive profits for a few years until market prices are driven down by the investment and competitors. The end result will be gigantic losses for the current investor towards the end of the simulation period. This is not acceptable unless the investor can sell his assets after 3-4 years and have another company take the losses.

Player = Coal Company, Competition = Low, Load Growth = Low

This simulation also starts with a high expectation of profits, and it may seem reasonable to begin investing at the start of the simulation. Since competition is low, the investor can nearly be confident that his investments will result in profits. However, since the load growth is 'Low', the investor should restrict the number of investments he makes as to not drive the market prices down due to excess capacity. The user can choose to invest at the beginning of the simulation but he will take some losses further into the game. In this scenario, it is better to let the market prices increase for a few years to gain the maximum profit of his investment at the end of the simulation. Otherwise, the low demand and excess capacity will drive market prices down and have the investor lose millions. The best strategy to maximize profits is to hold off investing until the market price increases near the end of the end of the simulation, and to invest in only one or two plants at most.

Player = Coal Company, Competition = Low, Load Growth = High

The user should expect massive profits from this scenario. Make use of the investment analysis option to maximize those profits.

Player = Gas Company, Competition = High, Load Growth = Low

Competitor gas companies will only come into play if the current market price is greater than their bidding price (and also yours). Otherwise, there will be no competition. The same is true for coal companies. This prevents the competitors from unfairly driving down the market prices to drive other investors out of business. Since it is not optimal for you nor your competitors to invest gas generation in the system until the market price reaches 75 or 100 (this depends on the number of planning years), do not invest until the price forecast or the investment analysis shows this optimum. Since competition is high, expect your competitor to invest as soon as the optimum market price is reached. *For a 5 year planning period, this optimum is never reached.* In a 10 yr planning period, a gas company is optimum to invest near the end of the period. Since the market prices reach optimum near the end of the 10 yr simulation period, competition will not have much effect in driving down the market prices and investor profits. If the planning period is greater than 10 yrs, it is optimal to invest as soon as the market price can support your bidding price. Excess capacity will drive the market price below the $75/MWHR or 100 level and turn off competitors. This should result in a good profit for the investor.

Player = Gas Company, Competition = Low, Load Growth = Low

Similarly to the scenario above, it is also economical to wait until the market price meets your bidding price. Competitors will only invest in the system if the bidding price (same as yours) is less than the market price. To maximize profits, invest so that the market price is always slightly below 75 or 100 (depending on number of planning years). Under this scenario, the investor can expect to achieve windfall profits if he invests correctly. This is an interesting scenario because lack of investment in the system for quite a few years leads to high prices. This will result in high prices staying high for quite a few years even after investments take place because of the current low elasticity of demand in driving down market prices.

Player = Gas Company, Competition = Low, Load Growth = High

Similarly to the scenario above, it is also economical to wait until the market price meets your bidding price, but since the load growth is high, the investors should not have to wait very long to achieve profitable investments alternatives. Since competition is low in this scenario, it will result in windfall profits for the investor.

Strategies

Profit maximizing strategies require the study of the investment analysis for every planning period. Some investment alternatives are good investments if competition does not occur, or if done when the market price is high enough. For most simulations, investing around the highest load bus (Bus 5) will result in optimal profits. For example, if the maximum load in the system is on Bus 5, investing at Bus 5 would be putting the generation near the load and resulting in optimal dispatch of generation by the optimal power flow program and thus maximum profits. If competition is heavy and load growth is low, it may be only economical to invest when the planning period is short (~5 years). Gas company players bring very expensive generation to the system and should only invest when the market prices are high enough (and will stay high enough) to allow the investor to obtain a reasonable profit based on his high operating costs for the remainder of the simulation period. Investing for short-term profits often brings heavy losses in the long-term, especially if competition is heavy or load growth is low.

Some basic simulation hints are summarized below:

- Place generation near load
- Always inspect the Investment Alternative Analysis
- Net Present Value is not always the best judge of maximum profits
- Gas generation is expensive and can cause either windfall profits or heavy losses
- Competitors will always be similar to the investor
- Competitors will only invest if the market price is above the bidding price
- Bidding prices obtain reasonable profits and are above marginal cost
- Market prices are based on capacity vs demand
- Short-sighted investing can result in massive losses
- Load demand does not decrease based on prices but will influence the prices.

Known Issues

- Currently there is a bug with the optimal power flow. If the load exceeds the generation capacity, the optimal power flow will stretch the generation above capacity to obtain solvency in the power flow. This may occur after many years of high load growth and little investment.

- The dynamic color coding of the contingency analysis figure may sometimes overdraw the bus voltages if there is both a high and low limit violation at that bus.

- The program does not offer error catching when the user enters menu options. It is assumed the user will enter menu options correctly and not enter any special codes.

V. Conclusions

The power simulation project was definitely a challenge to achieve and took many days of development and testing. The results have been a success to the purpose of the project: to study the effects of investments, demand-price responsiveness, competition, and reliability in a power system. A small 9-bus system with 8 transmission lines was choosen as the test case. The program can further be modified to include a larger system. Furthermore, the program could also be upgraded to allow investments on new transmission lines in the system.

Two types of generation technologies were considered: coal steam and gas turbine. The user can choose the type of technology to study investments. The competitor in the simulation is always of the same type of the generation technology being studied. Coal steam plant investments were priced the same as existing generation in the system, whereas gas turbine was expensive and took years to reach a profitable investment. Coal steam technologies faired more amiable investment alternatives, however, the gas turbines allowed more possible windfalls in profits if the market prices were increased high enough.

Load on the system was modeled as the total demand. Supply was modeled as the total capacity of all the generation in the system. Scarcity, the ratio of demand and supply, determined market prices. If there was excess generation in the system, the market price would trend downward, and vice versa. However, the demand did not reduce in response to high prices, which could be an enhancement for the program. Load increased every year as a percentage of the load growth level setting (default low) applied at the beginning of the simulation. The load growth was distributed randomly to each bus throughout the system. Each bus contained a random load growth amount in each simulation year and a forecast of future load growth.

Investment alternatives were analyzed and printed to the simulation per the user's request. Each alternative considered adding a generator at a specific bus, the future load forecasts, the future optimal power flow based on the new load, and calculated yearly revenue for the investment alternative. This analysis, without consideration of competition, educated the user of possibile short-term profits, loses, and present value analysis.

Different scenarios have been applied to study varying load, competition, and investment on the system. Some conclusions have been made regarding the energy-only market system studied in the simulation. Under a highly competitive environment, market prices decrease towards marginal costs. Whereas in a less competitive system, investor market power becomes apparent in the increases in market prices, and windwall profits for the investor. A balance and diversity of generation is needed to keep market prices sensible and offer a reasonable return of investment (profit) for the investors involved in providing the energy for this electric power system. Furthermore, the decentralized generation planning studied in this simulation was inefficient for supplying energy to meet the demand.

VII. References

Resource Adequacy & Electricity Markets, *William W. Hogan*, ENERGYBIZ MAGAZINE September/October 2005

Supplying the Generation to Meet the Demand, *Alex D. Papalexopoulos*, IEEE power & energy magazine, July/august 2004

Long-Term Planning in Restructured Power Systems - Dynamic Modelling of Investments in New Power Generation under Uncertainty -*Audun Botterud*

U.S. Electric Utility Sales, Revenue and Average Retail Price of Electricity http://www.eia.doe.gov/cneaf/electricity/page/at_a_glance/sales_tabs.html

Power Systems Planning ECE 553, *Audun Bottered*, Assignment #1

MATPOWER, A MATLAB Power System Simulation Package, MATPOWER User's Manual *by Ray D. Zimmerman, Carlos E. Murillo-Sánchez & Deqiang (David) Gan*

Security-Constrained Optimal Power Flow With Contingency Case Screening And Base Case Simulations, *Kathleen E. Williams*, 12/15/2005, ECE557 Fault Tolerant Systems

VII. Appendices

Appendix A: Generation Technology Data
Appendix B: Default System Overview
Appendix C: Default System Data File
Appendix D: MATLAB CODE

Appendix A: Generation Technology Data

Coal Steam Plant: 5-YR Planning Period

Economic lifetime (planning period) in years: 5
Average heate rate BTU per kWh: 9800
Investment cost $ per kW: 1400
$/MBtu: 2
Fixed O&M Cost $ per kW per Year: 15
Variable O&M Cost $ per MWh: 5
Fuel Cost and O&M Escalation percent per year: 0.05
Discount Rate: 0.1
Fixed Charge Rate: 0.18
Capacity in MW: 50
Capacity Factor: 0.9
Capital Recovery Factor: 0.263797
Levelizing Factor for uniform inflation: 1.09492
Fixed Levelized Annual Costs $ per MW per YEAR: 268423.7
Variable Levelized Annual Costs $ per MWH : 26.93
Total Investment Cost $ Millions: 70
Yearly Operating Cost $ per MWH : 26.9
Yearly Operating Cost $ M per yr: 9.7

Coal Steam Plant: 10-YR Planning Period

Economic lifetime (planning period) in years: 10
Average heate rate BTU per kWh: 9800
Investment cost $ per kW: 1400
$/MBtu: 2
Fixed O&M Cost $ per kW per Year: 15
Variable O&M Cost $ per MWh: 5
Fuel Cost and O&M Escalation percent per year: 0.05
Discount Rate: 0.1
Fixed Charge Rate: 0.18
Capacity in MW: 50
Capacity Factor: 0.9
Capital Recovery Factor: 0.162745
Levelizing Factor for uniform inflation: 1.2108
Fixed Levelized Annual Costs $ per MW per YEAR: 270161.9
Variable Levelized Annual Costs $ per MWH : 29.79
Total Investment Cost $ Millions: 70
Yearly Operating Cost $ per MWH : 29.8
Yearly Operating Cost $ M per yr: 9.7

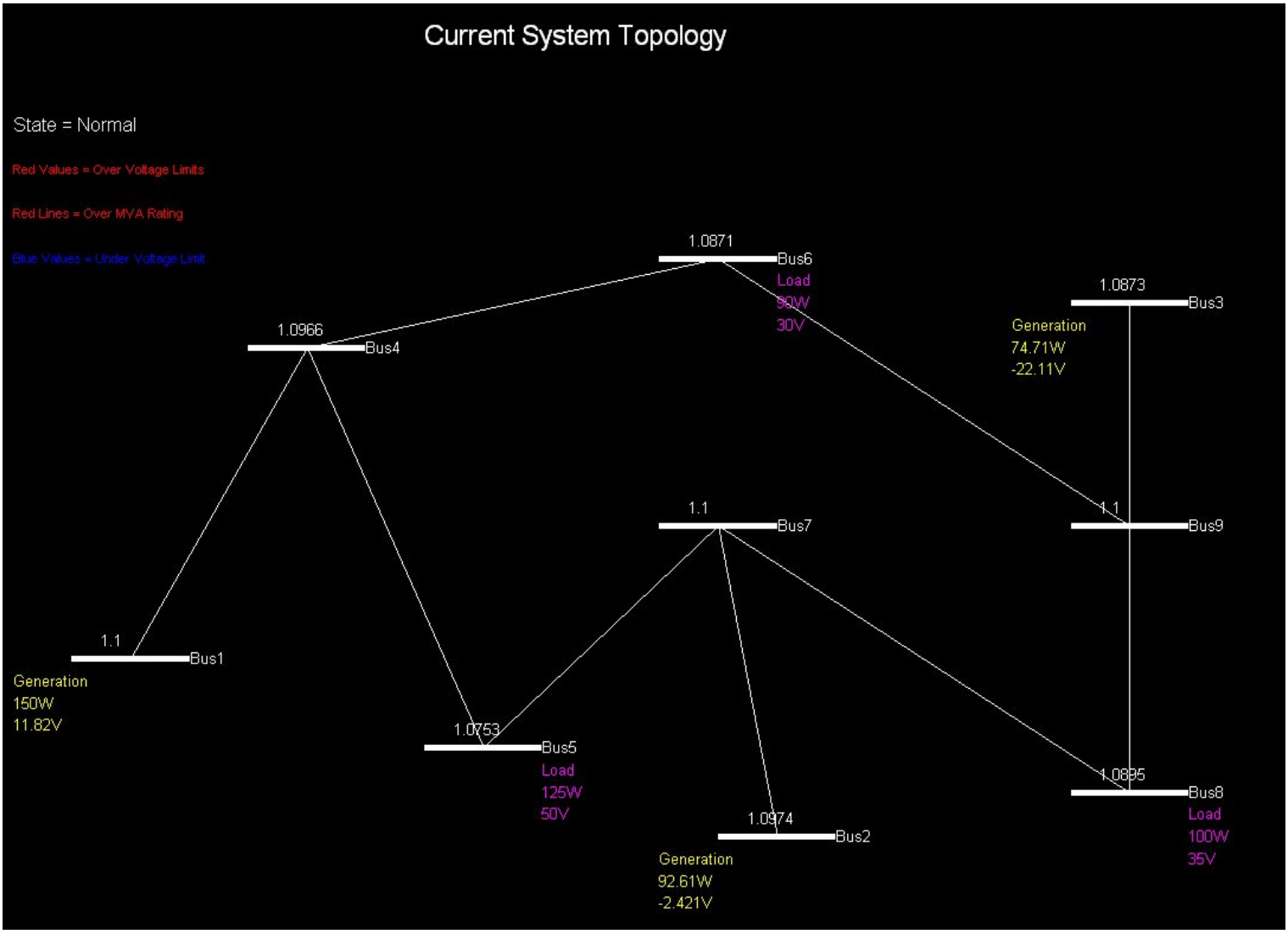
Gas Turbine Plant: : 5-YR Planning Period

Economic lifetime (planning period) in years: 5
Average heate rate BTU per kWh: 11800
Investment cost $ per kW: 350
$/MBtu: 3.5
Fixed O&M Cost $ per kW per Year: 1
Variable O&M Cost $ per MWh: 5
Fuel Cost and O&M Escalation percent per year: 0.08
Discount Rate: 0.1
Fixed Charge Rate: 0.18
Capacity in MW: 51
Capacity Factor: 0.9
Capital Recovery Factor: 0.263797
Levelizing Factor for uniform inflation: 1.15626
Fixed Levelized Annual Costs $ per MW per YEAR: 64156.26
Variable Levelized Annual Costs $ per MWH : 53.53
Total Investment Cost $ Millions: 17.9
Yearly Operating Cost $ per MWH : 53.5
Yearly Operating Cost $ M per yr: 18.6

Gas Turbine Plant: 10-YR Planning Period

Economic lifetime (planning period) in years: 10
Average heate rate BTU per kWh: 11800
Investment cost $ per kW: 350
$/MBtu: 3.5
Fixed O&M Cost $ per kW per Year: 1
Variable O&M Cost $ per MWh: 5
Fuel Cost and O&M Escalation percent per year: 0.08
Discount Rate: 0.1
Fixed Charge Rate: 0.18
Capacity in MW: 51
Capacity Factor: 0.9
Capital Recovery Factor: 0.162745
Levelizing Factor for uniform inflation: 1.36414
Fixed Levelized Annual Costs $ per MW per YEAR: 64364.14
Variable Levelized Annual Costs $ per MWH : 63.16
Total Investment Cost $ Millions: 17.9
Yearly Operating Cost $ per MWH : 63.2
Yearly Operating Cost $ M per yr: 18.6

Appendix B: Default System Overview



Appendix C: Default System Data File

```
function [baseMVA, bus, gen, branch, area, gencost] = wscc9bus
%WSCC9BUS    Defines the power flow data in a format similar to PTI.
%   [baseMVA, bus, gen, branch, area, gencost] = wscc9bus
%   The format for the data is similar to PTI format except where noted.
%   An item marked with (+) indicates that it is included in this data
%   but is not part of the PTI format. An item marked with (-) is one that
%   is in the PTI format but is not included here.
%
%   Bus Data Format
%       1   bus number (1 to 29997)
%       2   bus type
%               PQ bus          = 1
%               PV bus          = 2
%               reference bus   = 3
%               isolated bus    = 4
%       3   Pd, real power demand (MW)
%       4   Qd, reactive power demand (MVAR)
%       5   Gs, shunt conductance (MW (demanded?) at V = 1.0 p.u.)
%       6   Bs, shunt susceptance (MVAR (injected?) at V = 1.0 p.u.)
%       7   area number, 1-100
%       8   Vm, voltage magnitude (p.u.)
%       9   Va, voltage angle (degrees)
%   (-)     (bus name)
%       10  baseKV, base voltage (kV)
%       11  zone, loss zone (1-999)
%   (+) 12  maxVm, maximum voltage magnitude (p.u.)
%   (+) 13  minVm, minimum voltage magnitude (p.u.)
%
%   Generator Data Format
%       1   bus number
%   (-)     (machine identifier, 0-9, A-Z)
%       2   Pg, real power output (MW)
%       3   Qg, reactive power output (MVAR)
%       4   Qmax, maximum reactive power output (MVAR)
%       5   Qmin, minimum reactive power output (MVAR)
%       6   Vg, voltage magnitude setpoint (p.u.)
%   (-)     (remote controlled bus index)
%       7   mBase, total MVA base of this machine, defaults to baseMVA
%   (-)     (machine impedance, p.u. on mBase)
%   (-)     (step up transformer impedance, p.u. on mBase)
%   (-)     (step up transformer off nominal turns ratio)
%       8   status, 1 - machine in service, 0 - machine out of service
%   (-)     (% of total VARS to come from this gen in order to hold V at
%               remote bus controlled by several generators)
%       9   Pmax, maximum real power output (MW)
%       10  Pmin, minimum real power output (MW)
%
%   Branch Data Format
%       1   f, from bus number
%       2   t, to bus number
%   (-)     (circuit identifier)
%       3   r, resistance (p.u.)
%       4   x, reactance (p.u.)
%       5   b, total line charging susceptance (p.u.)
%       6   rateA, MVA rating A (long term rating)
%       7   rateB, MVA rating B (short term rating)
%       8   rateC, MVA rating C (emergency rating)
%       9   ratio, transformer off nominal turns ratio ( = 0 for lines )
%           (taps at 'from' bus, impedance at 'to' bus, i.e. ratio = Vf / Vt)
%       10  angle, transformer phase shift angle (degrees)
%   (-)     (Gf, shunt conductance at from bus p.u.)
%   (-)     (Bf, shunt susceptance at from bus p.u.)
%   (-)     (Gt, shunt conductance at to bus p.u.)
%   (-)     (Bt, shunt susceptance at to bus p.u.)
%       11  initial branch status, 1 - in service, 0 - out of service
%
% (+) Area Data Format
%       1   i, area number
%       2   price_ref_bus, reference bus for that area
%
% (+) Generator Cost Data Format
%       NOTE: If gen has n rows, then the first n rows of gencost contain
%       the cost for active power produced by the corresponding generators.
%       If gencost has 2*n rows then rows n+1 to 2*n contain the reactive
%       power costs in the same format.
%       1   model, 1 - piecewise linear, 2 - polynomial
%       2   startup, startup cost in US dollars
%       3   shutdown, shutdown cost in US dollars
%       4   n, number of cost coefficients to follow for polynomial
%           (or data points for piecewise linear) total cost function
%       5 and following, cost data, piecewise linear data as:
%                   x0, y0, x1, y1, x2, y2, ...
%           and polynomial data as, e.g.:
%                   c2, c1, c0
%           where the polynomial is c0 + c1*P + c2*P^2
%
% << this file modified [2005-Sep-02 13:51:27] by AJF >>

%%-----  Power Flow Data  -----%%
%% system MVA base
```

```
baseMVA = 100.0000;

%% bus data
bus = [
   1   3    0.0      0.0     0.0   0.0   1   1.0400   0.0000    16.5000   1   1.1000   0.9000;
   2   2    0.0      0.0     0.0   0.0   1   1.0250   9.3000    18.0000   1   1.1000   0.9000;
   3   2    0.0      0.0     0.0   0.0   1   1.0250   4.7000    13.8000   1   1.1000   0.9000;
   4   1    0.0      0.0     0.0   0.0   1   1.0260  -2.2000   230.0000   1   1.1000   0.9000;
   5   1  125.0     50.0     0.0   0.0   1   0.9960  -4.0000   230.0000   1   1.1000   0.9000;
   6   1   90.0     30.0     0.0   0.0   1   1.0130  -3.7000   230.0000   1   1.1000   0.9000;
   7   1    0.0      0.0     0.0   0.0   1   1.0260   3.7000   230.0000   1   1.1000   0.9000;
   8   1  100.0     35.0     0.0   0.0   1   1.0160   0.7000   230.0000   1   1.1000   0.9000;
   9   1    0.0      0.0     0.0   0.0   1   1.0320   2.0000   230.0000   1   1.1000   0.9000;
];

%% generator data
gen = [
   1   150.0000   50.0000   100.0000    -70.0000   1.0250   100.0000   1   150.0000   10.0000;
   2   150.0000   50.0000   100.0000    -70.0000   1.0250   100.0000   1   150.0000   10.0000;
   3   150.0000   50.0000   100.0000    -70.0000   1.0250   100.0000   1   150.0000   10.0000;
];


%% branch data
branch = [
   1   4   0.0000   0.0576   0.0000   250.0000   250.0000   250.0000   0.0000   0.0000   1;
   2   7   0.0000   0.0625   0.0000   250.0000   250.0000   250.0000   0.0000   0.0000   1;
   3   9   0.0000   0.0586   0.0000   300.0000   300.0000   300.0000   0.0000   0.0000   1;
   4   5   0.0100   0.0850   0.1760   250.0000   250.0000   250.0000   0.0000   0.0000   1;
   4   6   0.0170   0.0920   0.1580   250.0000   250.0000   250.0000   0.0000   0.0000   1;
   5   7   0.0320   0.1610   0.3060   250.0000   250.0000   250.0000   0.0000   0.0000   1;
   6   9   0.0390   0.1700   0.3580   150.0000   150.0000   150.0000   0.0000   0.0000   1;
   7   8   0.0085   0.0720   0.1490   250.0000   250.0000   250.0000   0.0000   0.0000   1;
   8   9   0.0119   0.1008   0.2090   150.0000   150.0000   150.0000   0.0000   0.0000   1;
];

%%-----  OPF Data  -----%%
%% area data
area = [
   1   5;
];
% Annual Costs for Coal Steam Plants

gencost = [
   2   0   0.00   3   0  50   0;
   2   0   0.00   3   0  50   0;
   2   0   0.00   3   0  50   0;
];


return;
```

Appendix D: MATLAB CODE

```
function [VoltageViolations] = bctgcalculateVoltageViolations9bus(V,bus,f,t)
%BCTGCALCULATEVOLTAGEVIOLATIONS - Calculates voltage violations for branch
% outages
%
% Inputs: 1P1Q estimated voltage V, bus data, from bus, to bus
% Outputs: Condensed matrix with ONLY violated bus voltage
% violations
%
% bctgCombinedVoltageViolations: (NX9) where N is the number of violations
% [<---branch contingency --->  <---bus violated--><-----------------under voltage--------------><--------over
voltage--------------------------->
% [from_bus             to_bus       bus             current_value low_limit percent_undervoltage  current_value
high_limit  percent_overvoltage]



Vm = abs(V);
%-------------------------------------------------------------------------
% Compute Buses Overvoltage
%-------------------------------------------------------------------------
VoltageViolations = zeros(9,7);

% Undervoltage Violations
for i=1:9
    if Vm(i,1) < bus(i,13)
            VoltageViolations(i,1) = i;
            VoltageViolations(i,2) = Vm(i,1);
            VoltageViolations(i,3) = bus(i,13);
            VoltageViolations(i,4) = Vm(i,1)./bus(i,13);
    else
    end;
end;

%OverVoltage Violations
for i=1:9
    if Vm(i,1) > bus(i,12)
      VoltageViolations(i,1) = i;
      VoltageViolations(i,5) = Vm(i,1);
      VoltageViolations(i,6) = bus(i,12);
      VoltageViolations(i,7) = Vm(i,1)./bus(i,12);
    else
    end;
end;

% Because MATLAB math is not so correct this is needed to correct it at limits
for i=1:9
    if abs(Vm(i,1) - bus(i,12)) < 0.00000005
      VoltageViolations(i,1) = 0;
    else
    end;
end;

%Eliminate zero rows
n=9;
while n >0
    if VoltageViolations(n,1) == 0
        VoltageViolations(n,:) = [];
    else
    end;
    n = n-1;
end;

VoltageViolations;

D = size(VoltageViolations);
VoltageViolations2 = zeros(D(1),9);
for i=1:D(1)
    VoltageViolations2(i,1) = f;
    VoltageViolations2(i,2) = t;
    VoltageViolations2(i,3) = VoltageViolations(i,1);
    VoltageViolations2(i,4) = VoltageViolations(i,2);
    VoltageViolations2(i,5) = VoltageViolations(i,3);
    VoltageViolations2(i,6) = VoltageViolations(i,4);
    VoltageViolations2(i,7) = VoltageViolations(i,5);
    VoltageViolations2(i,8) = VoltageViolations(i,6);
    VoltageViolations2(i,9) = VoltageViolations(i,7);
end;

VoltageViolations = VoltageViolations2;
clear VoltageViolations2;
return;




function playsimulation(player,comp,loadgrowth,N)
```

```
format short g;
OPTIONS.Hessian='on';
OPTIONS.GradObj='on';
clc
fprintf('\n');
fprintf('\n');
fprintf('\n Initializing Variables.... ');
fprintf('\n');
fprintf('\n');
pause(1);
clc;
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%------------------------------------------------------------------------
%                                  Setup
%------------------------------------------------------------------------
warning off MATLAB:singularMatrix;          % supresses error messages
clc;
% bus coordinates: bus x1 x2 y1 y2  used for making figure
bxy = [
1 -20 0 70 70;
2 90 110 30 30;
3 150 170 150 150;
4 10 30 140 140;
5 40 60 50 50;
6 80 100 160 160;
7 80 100 100 100;
8 150 170 40 40;
9 150 170 100 100;
];

profits = 0;
historicalprice = [
2005    69;];

% System Default Topology
[baseMVA, bus, gen, branch, area, gencost] = wscc9bus;
gen_old = gen;
bus_old = bus;
branch_old = branch;
gencost_old = gencost;

% Starting Year
startyear = [2006];
year = startyear;




if strcmp(player,'Coal Company') == 1

    % Annual Costs for Coal Steam Plants
    MW_ADD = 50; % this can be customized later
    [mc] = coalsteam(N,MW_ADD);% this is the marginal cost of the player's generation plan
    cs = mc;
    bidprice = 50; % this can be customized later
else
    % Annual Costs for Gas Turbine Plants
    MW_ADD = 51; % this can be customized later
    [mc] = gasturbine(N,MW_ADD);% this is the marginal cost of the player's generation plan
    bidprice = 65;
    if N <= 5
     bidprice = 65;
    else
     if N <=10
              bidprice = 75;
     else
         if N >= 10
             bidprice = 100;
         else
         end;
     end;
    end;
    cs = mc;
end;

%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%------------------------------------------------------------------------
%                                  Year Matrix
%------------------------------------------------------------------------
startyear2 = zeros(1,N);
startyear2(1,1) = startyear;
for k=2:N
    startyear2(1,k) = startyear + k-1;
end;
year = startyear2;
clear startyear2;
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


fprintf('\n');
fprintf('\n');
```

```
fprintf('\n Running Base Case and Unit Commitment Optimal Power Flow.... ');
fprintf('\n');
fprintf('\n');
pause(1);
clc;

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%----------------------------------------------------------------------------
%                                 Unit Decommitment Case
%----------------------------------------------------------------------------

options = mpoption('PF_ALG',1,'OPF_ALG',520,'OUT_GEN',1,'OUT_BRANCH',1,'VERBOSE',0);
[baseMVA, bus, gen, branch, success] = runuopf('wscc9bus',options);
branch = success;
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


clc
fprintf('\n');
fprintf('\n');
fprintf('\n Generating Random Load Growth Profile.... ');
fprintf('\n');
fprintf('\n');
pause(1);
clc;



%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%----------------------------------------------------------------------------
%                                 Forecast Load
%----------------------------------------------------------------------------

loadforecast = [];
D = size(bus_old);
busf = bus_old(1:D(1),3);
for k=1:N
 loadforecast2 = forecastload(busf,loadgrowth);
 busf = loadforecast2;
 loadforecast = [loadforecast loadforecast2];
end;
clear D, clear loadforecast2, clear busf, clear k;
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
clc
fprintf('\n');
fprintf('\n');
fprintf('\n Forecasting Price without Generation Investments........ ');
fprintf('\n');
fprintf('\n');
pause(1);
clc;

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%----------------------------------------------------------------------------
%                                 Forecast Price
%----------------------------------------------------------------------------
currentprice = historicalprice(1,2);
[priceforecast] =  priceforecaster(currentprice,loadforecast,bus_old,gen_old);
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


clc
fprintf('\n');
fprintf('\n');
fprintf('\n Running Contingency Screening and Analysis........ ');
fprintf('\n');
fprintf('\n');
pause(1);
clc;

%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
%============================================================================
% Contingency Case Screening
%============================================================================
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
clc
fprintf('\n');
fprintf('\n');
fprintf('\n N-1 Branch Outage Analysis ........ ');
fprintf('\n');
fprintf('\n');
pause(1);
clc

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%----------------------------------------------------------------------------
%                          Branch Contingency Screening
%----------------------------------------------------------------------------
[bctgCombinedMVAoverloads,bctgCombinedVoltageViolations] = bctgscreening9bus(baseMVA, bus, gen, branch)
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
clc
fprintf('\n');
fprintf('\n');
fprintf('\n N-1 Generator Outage Analysis ........ ');
fprintf('\n');
```

```
fprintf('\n');
pause(1);
clc

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%-------------------------------------------------------------------------
%                        Generator Contingency Screening
%-------------------------------------------------------------------------
[gctgCombinedMVAoverloads,gctgCombinedVoltageViolations] = gctgscreening9bus(baseMVA, bus, gen, branch)
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

clc
fprintf('\n');
fprintf('\n');
fprintf('\n Done Simulating!  ........ ');
fprintf('\n');
fprintf('\n');
pause(2);
clc



choice = 1;
while ~isequal(choice,-1)
fprintf('\n');
fprintf('\n');
fprintf('\n_____________________________________________________________________________');
fprintf('\n');
fprintf('\n               Simulation                                                  ');
fprintf('\n');
fprintf('\n   Player:  %s',player);
fprintf('\n                              Total Profits: %s',int2str(profits));
fprintf('\n                                       Year: %s',int2str(startyear));
fprintf('\n');
fprintf('\n');
fprintf('\n 1.  View Current System Topology (Figure)                 ');
fprintf('\n 2.  View Current Load Growth Forecast (Figure)          ');
fprintf('\n 3.  View Current Annual Price Forecast (Figure)          ');
fprintf('\n 4.  View Current System Topology with N-1 Contingency Analysis (Figure)         ');
fprintf('\n 5.  View Base Case and Unit Commitment Optimal Power Flow        ');
fprintf('\n 6.  View Investment Alternatives Analysis          ');
fprintf('\n 7.  Invest Generation and Move to Next Year       ');
fprintf('\n 8.  Do Not Invest Generation and Move to Next Year  ');
fprintf('\n 9.  Quit Simulation and Exit to Main Menu       ');
fprintf('\n');
fprintf('\n');


choice = input('Enter Choice Number:');
fprintf('\n');


fprintf('\n');
    if choice == 1
    % View System Topology (Figure)
    clc
    clf
    cla
    createnormal(bxy,branch,bus,gen)
    clc
    else
    end;
    if choice == 2
    % Plot Load Forecast
    clc
    clf
    cla
    plotloadforecast(year,loadforecast);
    clc
    else
    end;
    if choice == 3
    % Plot Price Forecast
    clf
    cla
    plotpriceforecast(year,priceforecast);
    clc
    else
    end;
    if choice == 4
    % Plot N-1 Contingency
    clc
    clf
    cla

printctg(bctgCombinedMVAoverloads,bctgCombinedVoltageViolations,gctgCombinedMVAoverloads,gctgCombinedVoltageViolations)
;

createctg(bxy,branch,bus,gen,bctgCombinedMVAoverloads,bctgCombinedVoltageViolations,gctgCombinedMVAoverloads,gctgCombin
edVoltageViolations)
    else
    end;
    if choice == 5
     % Execute Unit Commitment Optimal Power Flow
```

```
     runuopfspecial(baseMVA, bus, gen, branch, area, gencost,options);
    else
    end;
    if choice == 6
     % Analyze and View Investment Alternatives

analyzeinvestments(N,MW_ADD,bidprice,mc,priceforecast,loadforecast,year,baseMVA,bus_old,gen_old,branch_old,area,gencost
_old);
    else
    end;
    if choice == 7
      % Invest in New Generation
      [bus_old,gen_old,gencost_old,N] = invest(bus_old,gen_old,gencost_old,player,MW_ADD,bidprice,N,loadforecast);

      % Competition
      [bus_old,gen_old,gencost_old] =
competition(currentprice,comp,bus_old,gen_old,gencost_old,player,bidprice,loadforecast);

      % Go to Next Time Period

playsimulationNEXT(profits,startyear,player,comp,loadgrowth,N,bus_old,gen_old,gencost_old,MW_ADD,bidprice,mc,priceforec
ast,loadforecast,year,baseMVA,branch_old,area)

      % Exit
      choice = -1;
      clc
    else
    end;
    if choice == 8
      % Do not Invest and Move to N = N-1;
      N = N-1;

      % Competition
      [bus_old,gen_old,gencost_old] =
competition(currentprice,comp,bus_old,gen_old,gencost_old,player,bidprice,loadforecast);


      % Update the bus load for next year only
      bus_old(1:9,3) = loadforecast(1:9,1);

      % Go to Next Time Period

playsimulationNEXT(profits,startyear,player,comp,loadgrowth,N,bus_old,gen_old,gencost_old,MW_ADD,bidprice,mc,priceforec
ast,loadforecast,year,baseMVA,branch_old,area)

      % Exit
      choice = -1;
      clc
    else
    end;
    if choice == 9
      choice = -1;
      clc
    else
    end;
end;
return;




function
playsimulationFINAL(profits,startyear,player,comp,loadgrowth,N,bus_old,gen_old,gencost_old,MW_ADD,bidprice,mc,pricefore
cast,loadforecast,year,baseMVA,branch_old,area)
format short g;
clc
fprintf('\n');
fprintf('\n');
fprintf('\n Final Profit is: %s',num2str(profits));
fprintf('\n');
fprintf('\n');
pause(5);
clc;

return;







function howtoplay(r)

clc;
choice = 1;
while ~isequal(choice,-1)

fprintf('\n');
fprintf('\n');
    fprintf('\n_____________________________________________________________________');
    fprintf('\n');
    fprintf('\n                How to Play                                          ');
```

```
fprintf('\n');
fprintf('\n');
    fprintf('\n The default player is the Coal Company, but may choose to be a Gas Company.       ');
    fprintf('\n As a genco, your objective is to maximize your profit earning potential by investing          ');
    fprintf('\n and knowing where and when to invest your generation. The Power Sim Investor program         ');
    fprintf('\n will forecast the load growth at each bus on the system. A demand-side price response ');
    fprintf('\n algorithm will forecast a price that depends on the adequacy of the entire system. ');
    fprintf('\n Excess generation that does not serve load will bring the market price down and vice versa. ');
    fprintf('\n The Unit Decommitment Optimal power flow algorithm will determine which generators and how ');
    fprintf('\n much generation to dispatch based on price and location of the generation in order to meet ');
    fprintf('\n the load growth on each bus.');
    fprintf('\n');
fprintf('\n');
fprintf('\n Game Hints: The existing generators are coal plants (three) and will be bidding the same price as the Coal
');
fprintf('\n Company. However, if you are playing the Gas company, beware that your investment may not be dispatched');
    fprintf('\n because it expensive and coal generation may be in excess. When looking at investment alternatives, it
is ');
fprintf('\n more profitable to consider short term if competition is high and long-term if competition is low.');
fprintf('\n To maximize profit: Player = Coal Company, Competition = OFF, Load Growth = HIGH');
fprintf('\n ');
fprintf('\n Good luck and try to get positive profits!');
fprintf('\n');

fprintf('\n');
fprintf('\n');

      pchoice = input(' Exit to Main Menu (Y/N)','s');
      fprintf('\n');
      fprintf('\n');
fprintf('\n');
      if strcmp('Y',pchoice) || strcmp('y',pchoice)
        choice = -1;
      else
      end;
      clc;
end;


return;
```

```
function [gctgCombinedMVAoverloads,gctgCombinedVoltageViolations] = gctgscreening9bus(baseMVA, bus, gen, branch)
%GCTGSCREENING - Screening routine to examine each generator unit outage in the
% 9 bus system (total of 3 generators). Estimates all post-contingency voltage magnitudes
% (total of 9 buses) and all post-contingency MVA branch flows (total of 6 lines).
%
% Inputs: Base case AC power flow solutions with baseMVA, bus, gen, and
% branch matrices for the 9 bus system
% Outputs: Condensed matrix with ONLY violated branch MVA overloads and bus voltage
% violations
%
% gctgCombinedMVAoverloads: (NX4 where N is the number of violations
% [<-generator contingency --->  <---branch overloaded-->  <---violation--->
% [     generator bus               from_bus         to_bus   percent_overload]
%
% gctgCombinedVoltageViolations: (NX8) where N is the number of violations
% [<---generator contingency --->  <---bus violated--><-----------------under voltage--------------><--------over
voltage-------------------------->
% [generator bus                       bus               current_value low_limit percent_undervoltage  current_value
high_limit  percent_overvoltage]


%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
%=========================================================================
% Generator Contingency Case Screening
%=========================================================================
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

warning off MATLAB:singularMatrix;          % supresses error messages

%=========================================================================
% Global Variable Definitions
%=========================================================================
```

```
B = size(gen);                      % size of gen matrix
B1 = size(bus);                     % size of bus matrix
i = 1;                              % starts the contingency gen evaluation index
CombinedMVAoverloads = [];          % matrix to hold the gen contingency MVA overload list
CombinedVoltageViolations = [];     % matrix to hold the gen contingency voltage severity
%Form Dynamic Generator Contingency List
gendone = zeros(B(1),2);
gendone(1:B(1),1) = gen(1:B(1),1);
gendone(1:B(1),2) = gen(1:B(1),2);
gendone;
temp = ones(1,B(1))';
for k=1:B(1)
    temp(k,1) = k;
end;
genctglist = [gendone temp];

% eliminate the swing generator (gen 1)
genctglist(1,:) = [];

%genctglist2 = gendone;
% Eliminate the slack bus from the contingency analysis
%D = size(genctglist);              % determines the size of the matrix
%for k=1:D(1)
%    if bus(genctglist(k,1),2) == 3
%       genctglist2(k,:) = [];
%    else
%    end;
%end;
%genctglist = genctglist2;


%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%---------------------------------------------------------------------------
%   Begin Generator Contingency Analysis
%---------------------------------------------------------------------------

D = size(genctglist);               % determines the size of the matrix

while i < D(1) + 1                  % Cycle through every contingency generator outage


%---------------------------------------------------------------------------
% Generator Outage - Eliminate generator
%---------------------------------------------------------------------------

gennew = gen;
genout1 = genctglist(i,1);
genout2 = genctglist(i,2);
busnew = bus;


% Eliminate the generator for post-contingency study

for k=1:B(1)
    if gen(k,1) == genctglist(i,1) && (gen(k,2) == genctglist(i,2)) && genctglist(i,3) == i;
    gennew(k,:) = [];
    else
    end
end;
% Change the bus type from a PV to a PQ bus
for k=1:B1(1)
    if (bus(k,1) == genctglist(i,1))
        busnew(k,2) = 1;
    else
    end;
end;


%---------------------------------------------------------------------------
% Run first iteration of Fast-Decoupled
%---------------------------------------------------------------------------

options = mpoption('PF_ALG',2,'PF_MAX_IT_FD',1,'VERBOSE',0);
[baseMVA, busnew, gennew, branch, success] = gctgrunpf(gennew,busnew,'wscc9bus',options);

%---------------------------------------------------------------------------
% Setup Base Case Complex Voltage
%---------------------------------------------------------------------------

Vm = busnew(:,8);                   % This is the voltage magnitude column
Va = busnew(:,9);                   % This is the column of voltage angle values
Va = Va.*pi/180;                  % Convert the bus angle to radians
V = Vm .* exp(sqrt(-1) * Va);    % This is the complex voltage

%---------------------------------------------------------------------------
% Set-Up B-Prime Matrix, Sbus, Ybus
%---------------------------------------------------------------------------
alg = 2; % BX Method
[Bp, Bpp] =  makeB(baseMVA, busnew, branch, alg);
Sbus = makeSbus(baseMVA, busnew, gennew);
[Ybus, Yf, Yt] = makeYbus(baseMVA, busnew, branch);

%---------------------------------------------------------------------------
% Compute Branch Flows
```

```
%--------------------------------------------------------------------------

[br, Sf, St] = ctgcomputebranchflows9bus(busnew,gennew,branch,V,Yf,Yt,baseMVA);

%--------------------------------------------------------------------------
% Calculate MVA Overloads
%--------------------------------------------------------------------------

[MVAoverloads] = gctgcalculateMVAoverloads9bus(genout1,branch,Sf,St);
CombinedMVAoverloads = [CombinedMVAoverloads; MVAoverloads];

%--------------------------------------------------------------------------
% Calculate Voltage Violations
%--------------------------------------------------------------------------

[VoltageViolations] = gctgcalculateVoltageViolations9bus(V,busnew,genout1);
CombinedVoltageViolations = [CombinedVoltageViolations; VoltageViolations];

i = i+1;                            % switch to next generator outage contingency

end;  % end of the while loop









% Saves the Ranking Lists for the branch contingency
gctgCombinedMVAoverloads = [];
gctgCombinedMVAoverloads = CombinedMVAoverloads;
gctgCombinedVoltageViolations = [];
gctgCombinedVoltageViolations = CombinedVoltageViolations;


gctgCombinedMVAoverloads;

%==========================================================================
% Ranking of Voltage Violations
%==========================================================================
% Rank by high voltage descending, then low voltage ascending

gctgCombinedVoltageViolations2 = gctgCombinedVoltageViolations;
D = size(gctgCombinedVoltageViolations);

for j=1:D(1)
for i=1:D(1)
    if i > 1
        if gctgCombinedVoltageViolations2(i,5) < gctgCombinedVoltageViolations2(i-1,5)
            gctgCombinedVoltageViolationstemp1a = gctgCombinedVoltageViolations2(i,1);
            gctgCombinedVoltageViolationstemp1b = gctgCombinedVoltageViolations2(i,2);
            gctgCombinedVoltageViolationstemp1c = gctgCombinedVoltageViolations2(i,3);
            gctgCombinedVoltageViolationstemp1d = gctgCombinedVoltageViolations2(i,4);
            gctgCombinedVoltageViolationstemp1e = gctgCombinedVoltageViolations2(i,5);
            gctgCombinedVoltageViolationstemp1f = gctgCombinedVoltageViolations2(i,6);
            gctgCombinedVoltageViolationstemp1g = gctgCombinedVoltageViolations2(i,7);
            gctgCombinedVoltageViolationstemp1h = gctgCombinedVoltageViolations2(i,8);
            gctgCombinedVoltageViolationstemp2a = gctgCombinedVoltageViolations2(i-1,1);
            gctgCombinedVoltageViolationstemp2b = gctgCombinedVoltageViolations2(i-1,2);
            gctgCombinedVoltageViolationstemp2c = gctgCombinedVoltageViolations2(i-1,3);
            gctgCombinedVoltageViolationstemp2d = gctgCombinedVoltageViolations2(i-1,4);
            gctgCombinedVoltageViolationstemp2e = gctgCombinedVoltageViolations2(i-1,5);
            gctgCombinedVoltageViolationstemp2f = gctgCombinedVoltageViolations2(i-1,6);
            gctgCombinedVoltageViolationstemp2g = gctgCombinedVoltageViolations2(i-1,7);
            gctgCombinedVoltageViolationstemp2h = gctgCombinedVoltageViolations2(i-1,8);
            gctgCombinedVoltageViolations2(i-1,1) = gctgCombinedVoltageViolationstemp1a;
            gctgCombinedVoltageViolations2(i-1,2) = gctgCombinedVoltageViolationstemp1b;
            gctgCombinedVoltageViolations2(i-1,3) = gctgCombinedVoltageViolationstemp1c;
            gctgCombinedVoltageViolations2(i-1,4) = gctgCombinedVoltageViolationstemp1d;
            gctgCombinedVoltageViolations2(i-1,5) = gctgCombinedVoltageViolationstemp1e;
            gctgCombinedVoltageViolations2(i-1,6) = gctgCombinedVoltageViolationstemp1f;
            gctgCombinedVoltageViolations2(i-1,7) = gctgCombinedVoltageViolationstemp1g;
            gctgCombinedVoltageViolations2(i-1,8) = gctgCombinedVoltageViolationstemp1h;
            gctgCombinedVoltageViolations2(i,1) = gctgCombinedVoltageViolationstemp2a;
            gctgCombinedVoltageViolations2(i,2) = gctgCombinedVoltageViolationstemp2b;
            gctgCombinedVoltageViolations2(i,3) = gctgCombinedVoltageViolationstemp2c;
            gctgCombinedVoltageViolations2(i,4) = gctgCombinedVoltageViolationstemp2d;
            gctgCombinedVoltageViolations2(i,5) = gctgCombinedVoltageViolationstemp2e;
            gctgCombinedVoltageViolations2(i,6) = gctgCombinedVoltageViolationstemp2f;
            gctgCombinedVoltageViolations2(i,7) = gctgCombinedVoltageViolationstemp2g;
            gctgCombinedVoltageViolations2(i,8) = gctgCombinedVoltageViolationstemp2h;
        else
        end;
    else
    end;
end;
end;

for j=1:D(1)
for i=1:D(1)
    if i > 1
```

```
        if gctgCombinedVoltageViolations2(i,8) > gctgCombinedVoltageViolations2(i-1,8)
            gctgCombinedVoltageViolationstemp1a = gctgCombinedVoltageViolations2(i,1);
            gctgCombinedVoltageViolationstemp1b = gctgCombinedVoltageViolations2(i,2);
            gctgCombinedVoltageViolationstemp1c = gctgCombinedVoltageViolations2(i,3);
            gctgCombinedVoltageViolationstemp1d = gctgCombinedVoltageViolations2(i,4);
            gctgCombinedVoltageViolationstemp1e = gctgCombinedVoltageViolations2(i,5);
            gctgCombinedVoltageViolationstemp1f = gctgCombinedVoltageViolations2(i,6);
            gctgCombinedVoltageViolationstemp1g = gctgCombinedVoltageViolations2(i,7);
            gctgCombinedVoltageViolationstemp1h = gctgCombinedVoltageViolations2(i,8);
            gctgCombinedVoltageViolationstemp2a = gctgCombinedVoltageViolations2(i-1,1);
            gctgCombinedVoltageViolationstemp2b = gctgCombinedVoltageViolations2(i-1,2);
            gctgCombinedVoltageViolationstemp2c = gctgCombinedVoltageViolations2(i-1,3);
            gctgCombinedVoltageViolationstemp2d = gctgCombinedVoltageViolations2(i-1,4);
            gctgCombinedVoltageViolationstemp2e = gctgCombinedVoltageViolations2(i-1,5);
            gctgCombinedVoltageViolationstemp2f = gctgCombinedVoltageViolations2(i-1,6);
            gctgCombinedVoltageViolationstemp2g = gctgCombinedVoltageViolations2(i-1,7);
            gctgCombinedVoltageViolationstemp2h = gctgCombinedVoltageViolations2(i-1,8);
            gctgCombinedVoltageViolations2(i-1,1) = gctgCombinedVoltageViolationstemp1a;
            gctgCombinedVoltageViolations2(i-1,2) = gctgCombinedVoltageViolationstemp1b;
            gctgCombinedVoltageViolations2(i-1,3) = gctgCombinedVoltageViolationstemp1c;
            gctgCombinedVoltageViolations2(i-1,4) = gctgCombinedVoltageViolationstemp1d;
            gctgCombinedVoltageViolations2(i-1,5) = gctgCombinedVoltageViolationstemp1e;
            gctgCombinedVoltageViolations2(i-1,6) = gctgCombinedVoltageViolationstemp1f;
            gctgCombinedVoltageViolations2(i-1,7) = gctgCombinedVoltageViolationstemp1g;
            gctgCombinedVoltageViolations2(i-1,8) = gctgCombinedVoltageViolationstemp1h;
            gctgCombinedVoltageViolations2(i,1) = gctgCombinedVoltageViolationstemp2a;
            gctgCombinedVoltageViolations2(i,2) = gctgCombinedVoltageViolationstemp2b;
            gctgCombinedVoltageViolations2(i,3) = gctgCombinedVoltageViolationstemp2c;
            gctgCombinedVoltageViolations2(i,4) = gctgCombinedVoltageViolationstemp2d;
            gctgCombinedVoltageViolations2(i,5) = gctgCombinedVoltageViolationstemp2e;
            gctgCombinedVoltageViolations2(i,6) = gctgCombinedVoltageViolationstemp2f;
            gctgCombinedVoltageViolations2(i,7) = gctgCombinedVoltageViolationstemp2g;
            gctgCombinedVoltageViolations2(i,8) = gctgCombinedVoltageViolationstemp2h;
        else
        end;
    else
    end;
end;
end;

gctgCombinedVoltageViolations = gctgCombinedVoltageViolations2;

%-------------------------------------------------------------------------
% End Generator Contingency Analysis
%-------------------------------------------------------------------------
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

return;







function [priceforecast] = priceforecaster(currentprice,loadforecast,bus,gen)

%-------------------------------------------------------------------------
% Determine Yearly MW Load
%-------------------------------------------------------------------------
D = size(loadforecast);
lfyear = zeros(1,D(2));
for k=1:D(2)
    sum = 0;
    for j=1:D(1)
     sum = sum + loadforecast(j,k);
    end;
    lfyear(1,k) = sum;
end;


%-------------------------------------------------------------------------
% Determine Scarcity Factors
%-------------------------------------------------------------------------
D = size(gen);
sum = 0;
for k=1:D(1)
   sum = sum + gen(k,2);
end;
totalgen = sum;
lfyear;
sf = lfyear/totalgen;
sf = sf';
price = currentprice;

%-------------------------------------------------------------------------
% Determine Price
%-------------------------------------------------------------------------

D = size(sf);
priceforecast = zeros(D(1),1);
for k=1:D(1)
```

```
% Increase Price if Demand exceeds Supply
if sf(k,1) >= 0.9 && sf(k,1) <= 0.95
    priceforecast(k,1) = price*1.1;
else
    if sf(k,1) > 0.95 && sf(k,1) <= 1.0
     priceforecast(k,1) = price*1.2;
    else
        if sf(k,1) > 1.0
            priceforecast(k,1) = price*1.5;
        else
            % Decrease Price if Surplus in Generation
            priceforecast(k,1) = price*0.95;
        end;
    end;
end;
price = priceforecast(k,1);
end;
return;



function plotpriceforecast(year,priceforecast)

D = size(year);
plot(year,priceforecast(1:D(2),1),'-mo','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',[.49 1
.63],'MarkerSize',6);
title('Annual Price Forecast - Without New Generation');
xlabel('Year');
ylabel('$/MWhr');

return;









function
playsimulationNEXT(profits,startyear,player,comp,loadgrowth,N,bus_old,gen_old,gencost_old,MW_ADD,bidprice,mc,priceforec
ast,loadforecast,year,baseMVA,branch_old,area)
format short g;
clc
fprintf('\n');
fprintf('\n');
fprintf('\n Initializing Variables.... ');
fprintf('\n');
fprintf('\n');
pause(1);
clc;

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%---------------------------------------------------------------------------
%                                   Setup
%---------------------------------------------------------------------------
warning off MATLAB:singularMatrix;          % supresses error messages
clc;
% bus coordinates: bus x1 x2 y1 y2  used for making figure
bxy = [
1 -20 0 70 70;
2 90 110 30 30;
3 150 170 150 150;
4 10 30 140 140;
5 40 60 50 50;
6 80 100 160 160;
7 80 100 100 100;
8 150 170 40 40;
9 150 170 100 100;
];


% Update current price
currentprice = priceforecast(1,1);

% Update Year
startyear = startyear+1;
year = startyear;


%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%---------------------------------------------------------------------------
%                                   Year Matrix
%---------------------------------------------------------------------------
startyear2 = zeros(1,N);
startyear2(1,1) = startyear;
for k=2:N
    startyear2(1,k) = startyear + k-1;
```

```
end;
year = startyear2;
clear startyear2;
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

fprintf('\n');
fprintf('\n');
fprintf('\n Running Base Case and Unit Commitment Optimal Power Flow.... ');
fprintf('\n');
fprintf('\n');
pause(1);
clc;

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%------------------------------------------------------------------------
%                                 Unit Decommitment Case
%------------------------------------------------------------------------

options = mpoption('PF_ALG',1,'OPF_ALG',520,'OUT_GEN',1,'OUT_BRANCH',1,'VERBOSE',0);
[baseMVA, bus, gen, branch, success] = runuopfspecial(baseMVA, bus_old, gen_old, branch_old, area,
gencost_old,options);
branch = success;
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


fprintf('\n');
fprintf('\n');
fprintf('\n  Determining Annual and Total Profits.... ');
fprintf('\n');
fprintf('\n');
pause(1);
clc;


%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%------------------------------------------------------------------------
%                                 Determining Yearly Profits
%------------------------------------------------------------------------
% Coal Steam will have 50 MW generator
% Gas generators will be 51 MW
% Determine Total Profits on all generators own by company
profits2 = determineprofits(player,mc,gen_old,gen,currentprice);
profits = profits + profits2;
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

clc
fprintf('\n');
fprintf('\n');
fprintf('\n Generating Random Load Growth Profile.... ');
fprintf('\n');
fprintf('\n');
pause(1);
clc;



%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%------------------------------------------------------------------------
%                                 Forecast Load
%------------------------------------------------------------------------

loadforecast = [];
D = size(bus_old);
busf = bus_old(1:D(1),3);
for k=1:N
 loadforecast2 = forecastload(busf,loadgrowth);
 busf = loadforecast2;
 loadforecast = [loadforecast loadforecast2];
end;
clear D, clear loadforecast2, clear busf, clear k;
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
clc
fprintf('\n');
fprintf('\n');
fprintf('\n Forecasting Price without Generation Investments........ ');
fprintf('\n');
fprintf('\n');
pause(1);
clc;

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%------------------------------------------------------------------------
%                                 Forecast Price
%------------------------------------------------------------------------

[priceforecast] =  priceforecaster(currentprice,loadforecast,bus_old,gen_old);
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


clc
fprintf('\n');
fprintf('\n');
fprintf('\n Running Contingency Screening and Analysis........ ');
fprintf('\n');
```

```
fprintf('\n');
pause(1);
clc;

%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
%========================================================================
% Contingency Case Screening
%========================================================================
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
clc
fprintf('\n');
fprintf('\n');
fprintf('\n N-1 Branch Outage Analysis ........ ');
fprintf('\n');
fprintf('\n');
pause(1);
clc

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%------------------------------------------------------------------------
%                        Branch Contingency Screening
%------------------------------------------------------------------------
[bctgCombinedMVAoverloads,bctgCombinedVoltageViolations] = bctgscreening9bus(baseMVA, bus, gen, branch)
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
clc
fprintf('\n');
fprintf('\n');
fprintf('\n N-1 Generator Outage Analysis ........ ');
fprintf('\n');
fprintf('\n');
pause(1);
clc

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%------------------------------------------------------------------------
%                        Generator Contingency Screening
%------------------------------------------------------------------------
[gctgCombinedMVAoverloads,gctgCombinedVoltageViolations] = gctgscreening9bus(baseMVA, bus, gen, branch)
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

clc
fprintf('\n');
fprintf('\n');
fprintf('\n Done Simulating!  ........ ');
fprintf('\n');
fprintf('\n');
pause(2);
clc

if N <=0
    % go to final display

playsimulationFINAL(profits,startyear,player,comp,loadgrowth,N,bus_old,gen_old,gencost_old,MW_ADD,bidprice,mc,pricefore
cast,loadforecast,year,baseMVA,branch_old,area)
    choice = -1;
else
    % continue
end;


choice = 1;
while ~isequal(choice,-1) && N >0
fprintf('\n');
fprintf('\n');
fprintf('\n_____________________________________________________________________________');
fprintf('\n');
fprintf('\n                Simulation                                                   ');
fprintf('\n');
fprintf('\n   Player:  %s',player);
fprintf('\n                                Total Profits $ M: %s',num2str(profits,4));
fprintf('\n                                            Year: %s',int2str(startyear));
fprintf('\n');
fprintf('\n');
fprintf('\n 1.  View Current System Topology (Figure)                  ');
fprintf('\n 2.  View Current Load Growth Forecast (Figure)          ');
fprintf('\n 3.  View Current Annual Price Forecast (Figure)          ');
fprintf('\n 4.  View Current System Topology with N-1 Contingency Analysis (Figure)         ');
fprintf('\n 5.  View Base Case and Unit Commitment Optimal Power Flow        ');
fprintf('\n 6.  View Investment Alternatives Analysis          ');
fprintf('\n 7.  Invest Generation and Move to Next Year       ');
fprintf('\n 8.  Do Not Invest Generation and Move to Next Year  ');
fprintf('\n 9.  Quit Simulation         ');
fprintf('\n');
fprintf('\n');


choice = input('Enter Choice Number:');
fprintf('\n');


fprintf('\n');
    if choice == 1
    % View System Topology (Figure)
    clc
    clf
```

```
    cla
    createnormal(bxy,branch,bus,gen)
    clc
    else
    end;
    if choice == 2
    % Plot Load Forecast
    clc
    clf
    cla
    plotloadforecast(year,loadforecast);
    clc
    else
    end;
    if choice == 3
    % Plot Price Forecast
    clf
    cla
    plotpriceforecast(year,priceforecast);
    clc
    else
    end;
    if choice == 4
    % Plot N-1 Contingency
    clc
    clf
    cla

printctg(bctgCombinedMVAoverloads,bctgCombinedVoltageViolations,gctgCombinedMVAoverloads,gctgCombinedVoltageViolations)
;

createctg(bxy,branch,bus,gen,bctgCombinedMVAoverloads,bctgCombinedVoltageViolations,gctgCombinedMVAoverloads,gctgCombin
edVoltageViolations)
    else
    end;
    if choice == 5
     % Execute Unit Commitment Optimal Power Flow
     runuopfspecial(baseMVA, bus_old, gen_old, branch_old, area, gencost_old,options);
    else
    end;
    if choice == 6
     % Analyze and View Investment Alternatives

analyzeinvestments(N,MW_ADD,bidprice,mc,priceforecast,loadforecast,year,baseMVA,bus_old,gen_old,branch_old,area,gencost
_old);
    else
    end;
    if choice == 7
      % Invest in New Generation
      [bus_old,gen_old,gencost_old,N] = invest(bus_old,gen_old,gencost_old,player,MW_ADD,bidprice,N,loadforecast);

      % Competition
      [bus_old,gen_old,gencost_old] =
competition(currentprice,comp,bus_old,gen_old,gencost_old,player,bidprice,loadforecast);

      % Go to Next Time Period

playsimulationNEXT(profits,startyear,player,comp,loadgrowth,N,bus_old,gen_old,gencost_old,MW_ADD,bidprice,mc,priceforec
ast,loadforecast,year,baseMVA,branch_old,area)

      % Exit
      choice = -1;
      clc
    else
    end;
    if choice == 8
      % Do not Invest and Move to N = N-1;
      N = N-1;

      % Competition
      [bus_old,gen_old,gencost_old] =
competition(currentprice,comp,bus_old,gen_old,gencost_old,player,bidprice,loadforecast);

      % Update the bus load for next year only
      bus_old(1:9,3) = loadforecast(1:9,1);

      % Go to Next Time Period

playsimulationNEXT(profits,startyear,player,comp,loadgrowth,N,bus_old,gen_old,gencost_old,MW_ADD,bidprice,mc,priceforec
ast,loadforecast,year,baseMVA,branch_old,area)

      % Exit
      choice = -1;
      clc
    else
    end;
    if choice == 9
      choice = -1;
      clc
    else
    end;
end;


return;
```

```
clear;
clc
player = 'Coal Company';
competition = 'Low';
loadgrowth = 'Low';
N=5;

fprintf('\n');
fprintf('\n');

    fprintf('\n=====================================================================');
    fprintf('\n|               Power Sim Investor v1.1                             |');
    fprintf('\n=====================================================================');
    fprintf('\n Description: A 9-bus system needs power planning. Play as an investor');
    fprintf('\n and try to maximize your profits before your competitors beat you to ');
    fprintf('\n meeting the load growth, or play as an RTO to deliver power at       ');
    fprintf('\n minimal cost and minimal lost with N-1 contingency planning.         ');


choice = 1;
while ~isequal(choice,-1)
fprintf('\n');
fprintf('\n');
    fprintf('\n_____________________________________________________________________');
    fprintf('\n');
    fprintf('\n                Main Menu                                            ');
fprintf('\n');
fprintf('\n');
    fprintf('\n 1. How to Play - First time users select this option         ');
    fprintf('\n 2. Change Settings - Choose Player, Select Difficulty          ');
    fprintf('\n 3. Play Simulation          ');
    fprintf('\n 4. Exit           ');
 fprintf('\n');
fprintf('\n');


choice = input('Enter Choice Number:');
fprintf('\n');
    if choice == 1
      howtoplay
    else
    end;
    if choice == 2
      [player,competition,loadgrowth,N] = changesettings(player,competition,loadgrowth,N);
      clc
    else
    end;
    if choice == 3
      playsimulation(player,competition,loadgrowth,N);
    else
    end;
    if choice == 4
      choice = -1;
    else
    clc
    end;

end;
clc
clear
```

```
function [bus,gen,gencost,N] = invest(bus,gen,gencost,player,capacity,bidprice,N,loadforecast);

businvest = 2; % default bus to invest at
clc;
choice = 1;
while ~isequal(choice,-1)
fprintf('\n');
fprintf('\n');
fprintf('\n_____________________________________________________________________');
fprintf('\n');
fprintf('\n              Invest in Generation                                   ');
fprintf('\n');
fprintf('\n   Player:  %s',player);
fprintf('\n');
fprintf('\n');
fprintf('\n 1.  Choose Bus Number  (Default is Bus 2)             ');
fprintf('\n 2.  View Your Plant Information           ');
fprintf('\n 3.  Continue.....          ');
fprintf('\n');

choice = input('Enter Choice Number:');
fprintf('\n');


fprintf('\n');
    if choice == 1

    clc
    choice2 = 1
    while ~isequal(choice2,-1)
        clc
        fprintf('\n');
        fprintf('\n');
        fprintf('\n Choose Bus Number  (Default is Bus 2)             ');
        fprintf('\n');
        fprintf('\n');
        choice2 = input('Enter Choice Number:');
        fprintf('\n');
        if choice2 <= 9 && choice2 >=1
            businvest = choice2;
            fprintf('\n');
            fprintf('\n');
            fprintf('\n   You will be investing at bus number:  %s',int2str(businvest));
            fprintf('\n');
            pause(2);
            clc
            choice2 = -1;
        else
        end;
    end;
    else
    end;


    if choice == 2
        clc
        if strcmp(player,'Coal Company') == 1
        [mc] = coalsteam(N,capacity);
        else
        [mc] = gasturbine(N,capacity);
        end;

    else
    end;

    if choice == 3
      clc
      % Add New Investment Generator
      gen_add = [businvest          capacity           50          50          -50        1.025          100
1          capacity           10];
      gen = [gen; gen_add];
      % Add New Investment Generator Cost Information
      gencost_add   =         [2            0            0            3            0        bidprice            0];
      gencost = [gencost; gencost_add];
      % Change bus type except the slack
      if businvest == 1
      % do nothing
      else
       bus(businvest,2) = 2;
      end;
      % Update load with next year's load forecast
      bus(1:9,3) = loadforecast(1:9,1);
      % Change N
      N = N+1;
      choice = -1;
    else
    end;

end;
```

```
return;

function [loadforecast] = forecastload(bus,level)
%forecastload - Given an input bus matrix, common in the program and a user
%defined setting level (low, medium, high, off) creates a matrix with the
%forecasted load for one time increment. loadforecast will be returned with
%a NX1 matrix where N is the number of busses in the system and the value
%is the previous MW load on the bus plus a random new MW number. A random
%matrix for N busses is formed and the numbers generate form a "weight" of
%the total load growth on the system. If the load growth is 3%, the total
%load of the system is summed. The random numbers at each bus is summed and
%then made a a percentage that ad up to one. Each bus then has that
%percentage multipled by the 3% times system MW times percentage at that
%bus. Only MW or active power is forecasted.
%
%

%--------------------------------------------------------------------------
% Determine Current Load
%--------------------------------------------------------------------------
D = size(bus);
mw = 0;
for k=1:D(1)
    mw = mw + bus(k,1);
end;

%--------------------------------------------------------------------------
% Randomize Load Growth Shares at each bus
%--------------------------------------------------------------------------
growthr = rand(9,1);
growth = 0;
for k=1:D(1)
    growth = growth + growthr(k,1);
end;
growth = (growthr/growth);

%--------------------------------------------------------------------------
% Determine Forecast Load @ Each Bus
%--------------------------------------------------------------------------

% Low 5%
if strcmp(level,'Low') == 1
loadforecast = bus(1:D(1),1) + growth*mw*0.05;
else
end;

% Medium 10%
if strcmp(level,'Medium') == 1
loadforecast = bus(1:D(1),1) + growth*mw*0.10;
else
end;

% High 20%
if strcmp(level,'High') == 1
loadforecast = bus(1:D(1),1) + growth*mw*0.20;
else
end;

% Default
if strcmp(level,'Off') == 1
loadforecast = bus(1:D(1),1);
else
end;

return;
```

```
function [profits] = determineprofits(player,mc,gen_old,gen,currentprice)

if strcmp(player,'Coal Company') == 1
D = size(gen)
sum = 0;
for k=1:D(1)
    if gen_old(k,2) == 50
       sum = sum + (gen(k,2)*currentprice*8760 - (mc(1,1)*gen_old(k,2)+ mc(1,2)*gen(k,2)*8760))/10^6
    else
    end;
end;

profits = sum;
else
D = size(gen)
sum = 0;
for k=1:D(1)
    if gen_old(k,2) == 51
       sum = sum + (gen(k,2)*currentprice*8760 - (mc(1,1)*gen(k,2)+ mc(1,2)*gen(k,2)*8760))/10^6
    else
    end;
end;

profits = sum;



end;
return;


function [bctgCombinedMVAoverloads,bctgCombinedVoltageViolations] = bctgscreening9bus(baseMVA, bus, gen, branch)
%BCTGSCREENING - Screening routine to examine each non-islanding branch outage in the 9 bus system
% (total of 49 branches). Estimates all post-contingency voltage magnitudes (total of 64 buses) and all
% post-contingency MVA branch flows (total of 77 branches)
%
% Inputs: Base case AC power flow solutions with baseMVA, bus, gen, and
% branch matrices for the ESCA 64 bus system
% Outputs: Condensed matrix with ONLY violated branch MVA overloads and bus voltage
% violations
%
% bctgCombinedMVAoverloads: (NX5) where N is the number of violations
% [<---branch contingency --->  <---branch overloaded-->  <---violation--->
% [from_bus             to_bus  from_bus         to_bus   percent_overload]
%
% bctgCombinedVoltageViolations: (NX9) where N is the number of violations
% [<---branch contingency --->  <---bus violated--><-----------------under voltage---------------><--------over
voltage---------------------------->
% [from_bus             to_bus       bus             current_value low_limit percent_undervoltage  current_value
high_limit  percent_overvoltage]


%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
%=========================================================================
% Branch Contingency Case Screening
%=========================================================================
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

warning off MATLAB:singularMatrix;          % supresses error messages

%=========================================================================
% Global Variable Definitions
%=========================================================================

branchctglist = [
   4 5;
   4 6;
   5 7;
   6 9;
   7 8;
   8 9;
];

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%-------------------------------------------------------------------------
%   Begin Branch Contingency Analysis
%-------------------------------------------------------------------------

D = size(branchctglist);            % determines the size of the matrix
B = size(branch);                   % size of branch matrix
i = 1;                              % starts the contingency branch evaluation index
f=0;                                % from bus variable, initially at zero
t=0;                                % to bus variable, initially at zero
CombinedMVAoverloads = [];          % matrix to hold the branch contingency MVA overload list
CombinedVoltageViolations = [];     % matrix to hold the branch contingency voltage severity

while i < D(1) + 1                  % Cycle through every contingency branch outage

%-------------------------------------------------------------------------
```

```
% Branch Outage - Eliminate branch
%----------------------------------------------------------------------

branchnew = branch;

% Eliminate the branch for post-contingency study

for k=1:B(1)
    if (branch(k,1) == branchctglist(i,1)) && (branch(k,2) == branchctglist(i,2))
    branchnew(k,:) = [];
    f = branchctglist(i,1);
    t = branchctglist(i,2);
    else
    end
end;

%----------------------------------------------------------------------
% Run first iteration of Fast-Decoupled (1P1Q)
%----------------------------------------------------------------------

options = mpoption('PF_ALG',2,'PF_MAX_IT_FD',1,'VERBOSE',0);
[baseMVA, bus, gen, branchnew, success] = bctgrunpf(branchnew,'wscc9bus',options);


%----------------------------------------------------------------------
% See the Complex Voltage without Branch Outage
%----------------------------------------------------------------------

Vm = bus(:,8);                  % This is the voltage magnitude column
Va = bus(:,9);                  % This is the column of voltage angle values
Va = Va.*pi/180;                % Convert the bus angle to radians
V = Vm .* exp(sqrt(-1) * Va);   % This is the complex voltage

%----------------------------------------------------------------------
% Set-Up B-Prime Matrix, Sbus, Ybus
%----------------------------------------------------------------------
alg = 2; % BX Method
[Bp, Bpp] =  makeB(baseMVA, bus, branchnew, alg);
Sbus = makeSbus(baseMVA, bus, gen);
[Ybus, Yf, Yt] = makeYbus(baseMVA, bus, branchnew);


%----------------------------------------------------------------------
% Compute Branch Flows
%----------------------------------------------------------------------

[br, Sf, St] = ctgcomputebranchflows9bus(bus,gen,branchnew,V,Yf,Yt,baseMVA);

%----------------------------------------------------------------------
% Calculate MVA Overloads
%----------------------------------------------------------------------

[MVAoverloads] = bctgcalculateMVAoverloads9bus(branchnew,Sf,St,f,t);
CombinedMVAoverloads = [CombinedMVAoverloads; MVAoverloads];

%----------------------------------------------------------------------
% Calculate Voltage Violations
%----------------------------------------------------------------------

[VoltageViolations] = bctgcalculateVoltageViolations9bus(V,bus,f,t);
CombinedVoltageViolations = [CombinedVoltageViolations; VoltageViolations];

i = i+1;                            % switch to next branch outage contingency

end;  % end of the while loop


% Saves the Lists for the branch contingency
bctgCombinedMVAoverloads = [];
bctgCombinedMVAoverloads = CombinedMVAoverloads;
bctgCombinedVoltageViolations = [];
bctgCombinedVoltageViolations = CombinedVoltageViolations;



%======================================================================
% Ranking of Voltage Violations
%======================================================================
% Rank by high voltage descending, then low voltage ascending

bctgCombinedVoltageViolations2 = bctgCombinedVoltageViolations;
D = size(bctgCombinedVoltageViolations);

for j=1:D(1)
for i=1:D(1)
    if i > 1
        if bctgCombinedVoltageViolations2(i,6) < bctgCombinedVoltageViolations2(i-1,6)
            bctgCombinedVoltageViolationstemp1a = bctgCombinedVoltageViolations2(i,1);
            bctgCombinedVoltageViolationstemp1b = bctgCombinedVoltageViolations2(i,2);
            bctgCombinedVoltageViolationstemp1c = bctgCombinedVoltageViolations2(i,3);
            bctgCombinedVoltageViolationstemp1d = bctgCombinedVoltageViolations2(i,4);
            bctgCombinedVoltageViolationstemp1e = bctgCombinedVoltageViolations2(i,5);
            bctgCombinedVoltageViolationstemp1f = bctgCombinedVoltageViolations2(i,6);
            bctgCombinedVoltageViolationstemp1g = bctgCombinedVoltageViolations2(i,7);
```

```
            bctgCombinedVoltageViolationstemp1h = bctgCombinedVoltageViolations2(i,8);
            bctgCombinedVoltageViolationstemp1i = bctgCombinedVoltageViolations2(i,9);
            bctgCombinedVoltageViolationstemp2a = bctgCombinedVoltageViolations2(i-1,1);
            bctgCombinedVoltageViolationstemp2b = bctgCombinedVoltageViolations2(i-1,2);
            bctgCombinedVoltageViolationstemp2c = bctgCombinedVoltageViolations2(i-1,3);
            bctgCombinedVoltageViolationstemp2d = bctgCombinedVoltageViolations2(i-1,4);
            bctgCombinedVoltageViolationstemp2e = bctgCombinedVoltageViolations2(i-1,5);
            bctgCombinedVoltageViolationstemp2f = bctgCombinedVoltageViolations2(i-1,6);
            bctgCombinedVoltageViolationstemp2g = bctgCombinedVoltageViolations2(i-1,7);
            bctgCombinedVoltageViolationstemp2h = bctgCombinedVoltageViolations2(i-1,8);
            bctgCombinedVoltageViolationstemp2i = bctgCombinedVoltageViolations2(i-1,9);
            bctgCombinedVoltageViolations2(i-1,1) = bctgCombinedVoltageViolationstemp1a;
            bctgCombinedVoltageViolations2(i-1,2) = bctgCombinedVoltageViolationstemp1b;
            bctgCombinedVoltageViolations2(i-1,3) = bctgCombinedVoltageViolationstemp1c;
            bctgCombinedVoltageViolations2(i-1,4) = bctgCombinedVoltageViolationstemp1d;
            bctgCombinedVoltageViolations2(i-1,5) = bctgCombinedVoltageViolationstemp1e;
            bctgCombinedVoltageViolations2(i-1,6) = bctgCombinedVoltageViolationstemp1f;
            bctgCombinedVoltageViolations2(i-1,7) = bctgCombinedVoltageViolationstemp1g;
            bctgCombinedVoltageViolations2(i-1,8) = bctgCombinedVoltageViolationstemp1h;
            bctgCombinedVoltageViolations2(i-1,9) = bctgCombinedVoltageViolationstemp1i;
            bctgCombinedVoltageViolations2(i,1) = bctgCombinedVoltageViolationstemp2a;
            bctgCombinedVoltageViolations2(i,2) = bctgCombinedVoltageViolationstemp2b;
            bctgCombinedVoltageViolations2(i,3) = bctgCombinedVoltageViolationstemp2c;
            bctgCombinedVoltageViolations2(i,4) = bctgCombinedVoltageViolationstemp2d;
            bctgCombinedVoltageViolations2(i,5) = bctgCombinedVoltageViolationstemp2e;
            bctgCombinedVoltageViolations2(i,6) = bctgCombinedVoltageViolationstemp2f;
            bctgCombinedVoltageViolations2(i,7) = bctgCombinedVoltageViolationstemp2g;
            bctgCombinedVoltageViolations2(i,8) = bctgCombinedVoltageViolationstemp2h;
            bctgCombinedVoltageViolations2(i,9) = bctgCombinedVoltageViolationstemp2i;
        else
        end;
    else
    end;
end;
end;

for j=1:D(1)
for i=1:D(1)
    if i > 1
        if bctgCombinedVoltageViolations2(i,9) > bctgCombinedVoltageViolations2(i-1,9)
            bctgCombinedVoltageViolationstemp1a = bctgCombinedVoltageViolations2(i,1);
            bctgCombinedVoltageViolationstemp1b = bctgCombinedVoltageViolations2(i,2);
            bctgCombinedVoltageViolationstemp1c = bctgCombinedVoltageViolations2(i,3);
            bctgCombinedVoltageViolationstemp1d = bctgCombinedVoltageViolations2(i,4);
            bctgCombinedVoltageViolationstemp1e = bctgCombinedVoltageViolations2(i,5);
            bctgCombinedVoltageViolationstemp1f = bctgCombinedVoltageViolations2(i,6);
            bctgCombinedVoltageViolationstemp1g = bctgCombinedVoltageViolations2(i,7);
            bctgCombinedVoltageViolationstemp1h = bctgCombinedVoltageViolations2(i,8);
            bctgCombinedVoltageViolationstemp1i = bctgCombinedVoltageViolations2(i,9);
            bctgCombinedVoltageViolationstemp2a = bctgCombinedVoltageViolations2(i-1,1);
            bctgCombinedVoltageViolationstemp2b = bctgCombinedVoltageViolations2(i-1,2);
            bctgCombinedVoltageViolationstemp2c = bctgCombinedVoltageViolations2(i-1,3);
            bctgCombinedVoltageViolationstemp2d = bctgCombinedVoltageViolations2(i-1,4);
            bctgCombinedVoltageViolationstemp2e = bctgCombinedVoltageViolations2(i-1,5);
            bctgCombinedVoltageViolationstemp2f = bctgCombinedVoltageViolations2(i-1,6);
            bctgCombinedVoltageViolationstemp2g = bctgCombinedVoltageViolations2(i-1,7);
            bctgCombinedVoltageViolationstemp2h = bctgCombinedVoltageViolations2(i-1,8);
            bctgCombinedVoltageViolationstemp2i = bctgCombinedVoltageViolations2(i-1,9);
            bctgCombinedVoltageViolations2(i-1,1) = bctgCombinedVoltageViolationstemp1a;
            bctgCombinedVoltageViolations2(i-1,2) = bctgCombinedVoltageViolationstemp1b;
            bctgCombinedVoltageViolations2(i-1,3) = bctgCombinedVoltageViolationstemp1c;
            bctgCombinedVoltageViolations2(i-1,4) = bctgCombinedVoltageViolationstemp1d;
            bctgCombinedVoltageViolations2(i-1,5) = bctgCombinedVoltageViolationstemp1e;
            bctgCombinedVoltageViolations2(i-1,6) = bctgCombinedVoltageViolationstemp1f;
            bctgCombinedVoltageViolations2(i-1,7) = bctgCombinedVoltageViolationstemp1g;
            bctgCombinedVoltageViolations2(i-1,8) = bctgCombinedVoltageViolationstemp1h;
            bctgCombinedVoltageViolations2(i-1,9) = bctgCombinedVoltageViolationstemp1i;
            bctgCombinedVoltageViolations2(i,1) = bctgCombinedVoltageViolationstemp2a;
            bctgCombinedVoltageViolations2(i,2) = bctgCombinedVoltageViolationstemp2b;
            bctgCombinedVoltageViolations2(i,3) = bctgCombinedVoltageViolationstemp2c;
            bctgCombinedVoltageViolations2(i,4) = bctgCombinedVoltageViolationstemp2d;
            bctgCombinedVoltageViolations2(i,5) = bctgCombinedVoltageViolationstemp2e;
            bctgCombinedVoltageViolations2(i,6) = bctgCombinedVoltageViolationstemp2f;
            bctgCombinedVoltageViolations2(i,7) = bctgCombinedVoltageViolationstemp2g;
            bctgCombinedVoltageViolations2(i,8) = bctgCombinedVoltageViolationstemp2h;
            bctgCombinedVoltageViolations2(i,9) = bctgCombinedVoltageViolationstemp2i;
        else
        end;
    else
    end;
end;
end;

bctgCombinedVoltageViolations = bctgCombinedVoltageViolations2;


%----------------------------------------------------------------------
% End Branch Contingency Analysis
%----------------------------------------------------------------------
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

return;


function [centroidbxy] = centroidlines(bxy)
```

```
%centroidlines - given an NX5 matrix with bus bars and coordinates,
%determines and returns the centroid or middle of the line.
%[bus xcoordinate1 xcoordinate2 ycoordinate1 ycoordinate2]

D = size(bxy);
centroidbxy = zeros(9,3);
for k=1:D(1)
centroidbxy(k,1) = bxy(k,1);
centroidbxy(k,2) = (bxy(k,2) + bxy(k,3))/2;
centroidbxy(k,3) = (bxy(k,4) + bxy(k,5))/2;
end

return;


function [player,competition,loadgrowth,N] = changesettings(player,competition,loadgrowth,N)
%changesettings - Changes player, competition level, loadgrowth level, and
%the planning period
format short g;

clc;
choice = 1;
while ~isequal(choice,-1)
fprintf('\n');
fprintf('\n');
    fprintf('\n_____________________________________________________________________________');
    fprintf('\n');
    fprintf('\n                Change Settings                                            ');
fprintf('\n');
fprintf('\n');
    fprintf('\n 1. Players          ');
    fprintf('\n 2. Competition Amongst Companies    ');
    fprintf('\n 3. Load Growth     ');
    fprintf('\n 4. N - Number of Years Planning  ');
    fprintf('\n 5. Exit to Main Menu         ');
fprintf('\n');
fprintf('\n');

choice = input('Enter Choice Number:');
fprintf('\n');
    if choice == 1
      clc
      fprintf('\n Players          ');
      fprintf('\n 1. Coal Company   ');
      fprintf('\n 2. Gas Company          ');
      fprintf('\n');
      fprintf('\n Current Player is the %s',player);
      fprintf('\n');
      pchoice = input(' Would you like to change Players? (Y/N)','s');
      fprintf('\n');
      if strcmp('Y',pchoice) || strcmp('y',pchoice)
        pchoice = input(' Choose the Player Number:');
        fprintf('\n');
            if pchoice == 1
                player = 'Coal Company';
            else
            end;
            if pchoice == 2
                player = 'Gas Company';
            else
            end;
      else
      end;
    fprintf('\n');
    fprintf('\n Current Player is the %s',player);
    pause(2)
    clc
    else
    end;
    if choice == 2
      clc
      fprintf('\n Competition          ');
      fprintf('\n 1. Off    ');
      fprintf('\n 2. Low          ');
      fprintf('\n 3. Medium          ');
      fprintf('\n 4. High          ');
      fprintf('\n');
      fprintf('\n Current Competition is set to %s',competition);
      fprintf('\n');
      pchoice = input(' Would you like to change the Competition setting (Y/N)','s');
      fprintf('\n');
      if strcmp('Y',pchoice) || strcmp('y',pchoice)
        pchoice = input(' Choose the Setting Number:');
        fprintf('\n');
            if pchoice == 1
                competition = 'Off';
            else
            end;
            if pchoice == 2
                competition = 'Low';
            else
            end;
            if pchoice == 3
                competition = 'Medium';
            else
```

```
            end;
            if pchoice == 4
                competition = 'High';
            else
            end;
        else
        end;
          fprintf('\n');
          fprintf('\n Current Competition Setting is on %s',competition);
          pause(2)
          clc
    else
    end;
    if choice == 3
      clc
      fprintf('\n Load Grwoth         ');
      fprintf('\n 1. Off    ');
      fprintf('\n 2. Low          ');
      fprintf('\n 3. Medium          ');
      fprintf('\n 4. High          ');
      fprintf('\n');
      fprintf('\n Current Growth is set to %s',loadgrowth);
      fprintf('\n');
      pchoice = input(' Would you like to change the Load Growth (Y/N)','s');
      fprintf('\n');
      if strcmp('Y',pchoice) || strcmp('y',pchoice)
        pchoice = input(' Choose the Setting Number:');
        fprintf('\n');
            if pchoice == 1
                loadgrowth = 'Off';
            else
            end;
            if pchoice == 2
                loadgrowth = 'Low';
            else
            end;
            if pchoice == 3
                loadgrowth = 'Medium';
            else
            end;
            if pchoice == 4
                loadgrowth = 'High';
            else
            end;
        else
        end;
          fprintf('\n');
          fprintf('\n Current Load Growth Setting is on %s',loadgrowth);
          pause(2)
          clc
    else
    end;
    if choice == 4
     clc
     fprintf('\n The Current Planning Period N is %s years',int2str(N));
     fprintf('\n');
     pchoice = input(' Would you like to change the Planning Period (Y/N)','s');
     if strcmp('Y',pchoice) || strcmp('y',pchoice)
       pchoice = input(' Enter the number of years (integer):');
       N = pchoice;
     else
     end;
       fprintf('\n');
       fprintf('\n The Current Planning Period N is %s years',int2str(N));
       pause(2)
       clc
    else
    end;
    if choice == 5
      choice = -1;
      clc
    else
    end;

end;
return;
```

```
function [mc] = coalsteam(N,capacity)

N;              % Economic lifetime (planning period) in years
ahr = 9800;     % Average heate rate BTU/kWh
ic = 1400;      % Investment cost $/kW
fc = 2.0;       % $/MBtu
FOM = 15;       % Fixed O&M Cost $/kW/Year
VOM = 5;        % Variable O&M Cost $/MWh
a = 0.05;        % Fuel Cost and O&M Escalation %/year
r = 0.10;       % Discount rate
FCR = 0.18;     % Fixed charge rate
```

```
capacity; % capacity in MW
capacityfactor = 0.9;
fprintf('\n   Coal Steam Plant:  ');
fprintf('\n'   );
fprintf('\n'   );
fprintf('\n   Economic lifetime (planning period) in years:  %s',int2str(N));
fprintf('\n   Average heate rate BTU per kWh:  %s',int2str(ahr));
fprintf('\n   Investment cost $ per kW:  %s',int2str(ic));
fprintf('\n   $/MBtu:  %s',num2str(fc,3));
fprintf('\n   Fixed O&M Cost $ per kW per Year:  %s',num2str(FOM,2));
fprintf('\n   Variable O&M Cost $ per MWh:  %s',int2str(VOM));
fprintf('\n   Fuel Cost and O&M Escalation percent per year:  %s',num2str(a,2));
fprintf('\n   Discount Rate:  %s',num2str(r,2));
fprintf('\n   Fixed Charge Rate:  %s',num2str(FCR,2));
fprintf('\n   Capacity in MW:  %s',int2str(capacity));
fprintf('\n   Capacity Factor:  %s',num2str(capacityfactor,2));


% Capital Recovery Factor
CRF = r*((1+r)^N)/((1+r)^N-1);
fprintf('\n   Capital Recovery Factor:  %s',num2str(CRF,6));
% Levelizing Factor for uniform inflation
LF = CRF*[1-(((1+a)/(1+r))^N)]/(r-a);
fprintf('\n   Levelizing Factor for uniform inflation:  %s',num2str(LF,6));
% Fixed Levelized Annual Costs
investment = ic*FCR*1000;               % Investment $/MW/YEAR
fixedom = FOM*LF*1000;                  % Fixed O&M $/MW/YEAR
k1 = investment + fixedom;              % Total $/MW/YEAR
fprintf('\n   Fixed Levelized Annual Costs $ per MW per YEAR:  %s',num2str(k1,7));
% Variable Levelized Annual Costs
fuel = ahr*fc*LF/1000;                  % Fuel $/MWh
VOMc = VOM*LF;                          % Var. O&M $/MWH
c1 = fuel + VOMc;                       % Total $/MWH
fprintf('\n   Variable Levelized Annual Costs $ per MWH :  %s',num2str(c1,4));
% Total Investment Cost
tic = (capacity*ic*10^3)/10^6;  % $ Millions
fprintf('\n   Total Investment Cost $ Millions:  %s',num2str(tic,3));
% Operating Costs
% Yearly Operating Cost
c1a = ((ahr*fc*1000)/10^6) +VOM;    % $/MWH
fprintf('\n   Yearly Operating Cost $ per MWH :  %s',num2str(c1,3));
% Per Year
yoc = capacity*capacityfactor*8760*c1a/10^6;   % $ M/yr
fprintf('\n   Yearly Operating Cost $ M per yr:  %s',num2str(yoc,3));
fprintf('\n');

mc = [k1 c1]; % Returns fixed and variable levelized annual costs
return;



function [bus,gen,gencost] = competition(currentprice,comp,bus,gen,gencost,player,bidprice,loadforecast);
capacity = 20;
compinvest = 0;

if currentprice >= bidprice

if strcmp(comp,'Off') == 1
    % There is not competition competition
else
end;

if strcmp(comp,'Low') == 1

       % < 25 percent or less chance there will be competiting activity
       num1 = rand;
       num2 = rand;

       if num1*num2 >=0.50
        % competition invests
        compinvest = 1;
       else
        % competition does not invest
       end;
else
end;

if strcmp(comp,'Medium') == 1

       % 50 percent or less chance there will be competiting activity
       num1 = rand;
       num2 = rand;

       if num1*num2 >=0.20
        % competition invests
        compinvest = 1;
       else
        % competition does not invest
       end;
else
end;

if strcmp(comp,'High') == 1

        % 80 percent or less chance there will be competiting activity
```

```
       num1 = rand;
       num2 = rand;

       if num1*num2 >=0.05
        % competition invests
        compinvest = 1;
       else
        % competition does not invest
       end;

 else
 end;


if compinvest == 1
    % Randomly determine on which bus the investment will be
    num1 = rand;
    num1 = floor(num1*10);
    if num1 == 0
        businvest = 9;
    else
        businvest = num1;
    end;
    fprintf('\n');
    fprintf('\n !!!! Attention, one of your competitors invested in new generation !!!!!');
    fprintf('\n');
    fprintf('\n Competitor will be investing 20 MW at bus: %s', int2str(businvest));
    fprintf('\n');
    fprintf('\n');
    pause(3);
    gen_add = [businvest          capacity           50          50          -50        1.025          100            1
capacity           10];
    gen = [gen; gen_add];
    % Add New Investment Generator Cost Information
    gencost_add   =         [2            0            0           3           0        bidprice            0];
    gencost = [gencost; gencost_add];
    % Change bus type except the slack
      if businvest == 1
      % do nothing
      else
       bus(businvest,2) = 2;
      end;
else
end;


else
end;

return;

function
createctg(bxy,branch,bus,gen,bctgCombinedMVAoverloads,bctgCombinedVoltageViolations,gctgCombinedMVAoverloads,gctgCombin
edVoltageViolations)
%createnormal    Creates Display of N-1 Contingency System Topology
%   Dynamically color codes based on bus voltage being too high, or too low
%   or MVA rating of line at either end of line being exceeded
%   Generation and Load is displayed on each bus per the input matrix
%   bxy holds the location of the bus bars (bus x1 x2 y1 y2) and is NX4
%   branch,bus,gen is an output as a result of the power flow solution
%   gctgCombinedMVAoverloads,gctgCombinedVoltageViolations
%   Matrices that tell of possible line overloads and bus voltage
%   violations as a result of any one generator outage
%   bctgCombinedMVAoverloads,bctgCombinedVoltageViolations
%   Matrices that tell of possible line overloads and bus voltage
%   violations as a result of any one non-islanding transmission line
%   outage
%-------------------------------------------------------------------------
% Variables

state = ' Contingency'; %current game state

% Sorts and Sums the Generation
gen = sortrows(gen,[1 2]);

D = size(gen);
gen2 = gen;
n = 2;
for k=2:D(1)
    if (gen(k,1) == gen(k-1,1))
       gen2(n,2) = gen2(n,2) + gen2(n-1,2);
       gen2(n,3) = gen2(n,3) + gen2(n-1,3);
       gen2(n-1,:) = [];
       n = n-1;
    else
    end;
    n = n+1;
end;

gen = gen2;


% Creates a general data matrx used in this procedure
D = size(bus);
G = size(gen);
```

```
busdata = zeros(D(1),7);
for k=1:D(1)
    busdata(k,1) = bus(k,1);
    busdata(k,2) = bus(k,8);
    busdata(k,3) = bus(k,9);
    busdata(k,4) = 0;
    busdata(k,5) = 0;
    for j =1:G(1)
        if bus(k,1) == gen(j,1)
            busdata(k,4) = gen(j,2);
            busdata(k,5) = gen(j,3);
        else
        end;
    end;
    busdata(k,6) = bus(k,3);
    busdata(k,7) = bus(k,4);
end;

% Creates a Matrix of Bus Labels
D = size(bus);
for k=1:D(1)
bustext(k,1:4) = strcat('Bus',int2str(bus(k,1)));
end;

% Since there are multiple generators with multiple contingencies, make
% sure only the worst are shown on top
gctgCombinedVoltageViolations = sortrows(gctgCombinedVoltageViolations,[2 8]);
D = size(gctgCombinedVoltageViolations);
gctgCombinedVoltageViolations2 = gctgCombinedVoltageViolations;
n = 2;
for k=2:D(1)
    if (gctgCombinedVoltageViolations(k,2) == gctgCombinedVoltageViolations(k-1,2)) &&
gctgCombinedVoltageViolations(k,5) == gctgCombinedVoltageViolations(k-1,5)
       gctgCombinedVoltageViolations2(n-1,:) = [];
       n = n-1;
    else
    end;
    n = n+1;
end;

gctgCombinedVoltageViolations = gctgCombinedVoltageViolations2;

bctgCombinedVoltageViolations = sortrows(bctgCombinedVoltageViolations,[3 6]);
D = size(bctgCombinedVoltageViolations);
bctgCombinedVoltageViolations2 = bctgCombinedVoltageViolations;
n = 2;
for k=2:D(1)
    if (bctgCombinedVoltageViolations(k,3) == bctgCombinedVoltageViolations(k-1,3) &&
bctgCombinedVoltageViolations(k,7) == bctgCombinedVoltageViolations(k-1,7))
       bctgCombinedVoltageViolations2(n-1,:) = [];
       n = n-1;
    else
    end;
    n = n+1;
end;
bctgCombinedVoltageViolations = bctgCombinedVoltageViolations2;




%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% Set the GRID
cla; % clear Figure Axis
clf; % clear Figure
set(0,'Units','normalized'); % normalizes the screen coordinate system to 0,0,1,1
axis([0 200 0 200])
axis off; %hides the axis
set(gcf,'Color','k') % sets the background of the figure to black
rect = [100,100,600,600];
set(gcf,'Position',rect) % sets the position of the figure
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% Print Current States and Scores
title1 = text(40,210,'Current System Topology','FontSize',17,'Color','w'); % Title
title1 = text(-30,195,strcat('State = ',state),'FontSize',12,'Color','w'); %Game State
title1 = text(-30,185,'Current System Possible N-1 Contingencies','FontSize',8,'Color','w'); %info
title1 = text(-30,180,'Red Values = Possible Over Voltage Limits','FontSize',8,'Color','r'); %info
title1 = text(-30,175,'Red Lines = Possible Over MVA Rating','FontSize',8,'Color','r'); %info
title1 = text(-30,170,'Blue Values = Possible Under Voltage Limit','FontSize',8,'Color','b'); %info
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% Prints Bus Bars and Adds IDs
D=size(bxy);
% prints the bus bars
for k=1:D(1)
createline(bxy(k,2),bxy(k,3),bxy(k,4),bxy(k,5),'w',4);
end;
% prints the bus bar text
for k=1:D(1)
createtext(bustext(k,1:4),bxy(k,2),bxy(k,3),bxy(k,4),bxy(k,5));
```

```
end;
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% Create Transmission Lines -  Dynamically color coded based on MVA overload
% Determine the Centroid of the bus bars
[branches] = centroidlines(bxy);

D=size(branch);

for k=1:D(1)
 createline(branches(branch(k,1),2),branches(branch(k,2),2),branches(branch(k,1),3),branches(branch(k,2),3),'w',1);
end;

G=size(bctgCombinedMVAoverloads);
if G(1) > 0
    for k=1:G(1)

createline(branches(bctgCombinedMVAoverloads(k,3),2),branches(bctgCombinedMVAoverloads(k,4),2),branches(bctgCombinedMVA
overloads(k,3),3),branches(bctgCombinedMVAoverloads(k,4),3),'r',1);
    end;
else
end;

H=size(gctgCombinedMVAoverloads);
if H(1) > 0
    for k=1:H(1)

createline(branches(gctgCombinedMVAoverloads(k,2),2),branches(gctgCombinedMVAoverloads(k,3),2),branches(gctgCombinedMVA
overloads(k,2),3),branches(gctgCombinedMVAoverloads(k,3),3),'r',1);
    end;
else
end;

%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% Print Voltages on Bus Bars - Dynamically Color Codes Values
% Blue = Undervoltage Limit, Red = Overvoltage Limit
% need to print contigency values

busvoltages = zeros(0,0);
D = size(bxy);
%for k=1:D(1)
%    text([branches(k,2)-5 branches(k,2)-5],[branches(k,3)+4 branches(k,3)+4],num2str(busdata(k,2)),'Color','w');
%end;

G = size(bctgCombinedVoltageViolations);
if G(1) > 0
    for k=1:G(1)
        if bctgCombinedVoltageViolations(k,9) > 1
            j = bctgCombinedVoltageViolations(k,3);
            text([branches(j,2)-5 branches(j,2)-5],[branches(j,3)+4
branches(j,3)+4],num2str(bctgCombinedVoltageViolations(k,7)),'Color','r');
        else
        end;
    end;
else
end;

H = size(gctgCombinedVoltageViolations);
if H(1) > 0
    for k=1:H(1)
        if gctgCombinedVoltageViolations(k,8) > 1
            j = gctgCombinedVoltageViolations(k,2);
            text([branches(j,2)-5 branches(j,2)-5],[branches(j,3)+4
branches(j,3)+4],num2str(gctgCombinedVoltageViolations(k,6)),'Color','r');
        else
        end;
    end;
else
end;

G = size(bctgCombinedVoltageViolations);
if G(1) > 0
    for k=1:G(1)
        if bctgCombinedVoltageViolations(k,6) < 1 &&  bctgCombinedVoltageViolations(k,6) > 0
            j = bctgCombinedVoltageViolations(k,3);
            text([branches(j,2)-5 branches(j,2)-5],[branches(j,3)+4
branches(j,3)+4],num2str(bctgCombinedVoltageViolations(k,4)),'Color','b');
        else
        end;
    end;
else
end;


H = size(gctgCombinedVoltageViolations);
if H(1) > 0
    for k=1:H(1)
        if gctgCombinedVoltageViolations(k,5) < 1 && gctgCombinedVoltageViolations(k,5) > 0
            j = gctgCombinedVoltageViolations(k,2);
            text([branches(j,2)-5 branches(j,2)-5],[branches(j,3)+4
branches(j,3)+4],num2str(gctgCombinedVoltageViolations(k,3)),'Color','b');
        else
```

```
        end;
    end;
else
end;



%-----------------------------------------------------------------------

%-----------------------------------------------------------------------
% Print Generation and Load
%D = size(busdata);
%for k=1:D(1)
%if busdata(k,4) > 0
%text([branches(k,2)-10 branches(k,2)-10],[branches(k,3)-5 branches(k,3)-5],'Generation','Color','y');
%text([branches(k,2)-10 branches(k,2)-10],[branches(k,3)-10 branches(k,3)-
10],strcat(num2str(busdata(k,4)),'W'),'Color','y');
%text([branches(k,2)-10 branches(k,2)-10],[branches(k,3)-15 branches(k,3)-
15],strcat(num2str(busdata(k,5)),'V'),'Color','y');
%else
%end;
%if busdata(k,6) > 0
%text([branches(k,2)-10 branches(k,2)-10],[branches(k,3)-5 branches(k,3)-5],'Load','Color','m');
%text([branches(k,2)-10 branches(k,2)-10],[branches(k,3)-10 branches(k,3)-
10],strcat(num2str(busdata(k,6)),'W'),'Color','m');
%text([branches(k,2)-10 branches(k,2)-10],[branches(k,3)-15 branches(k,3)-
15],strcat(num2str(busdata(k,7)),'V'),'Color','m');
%else
%end;
%end;
%-----------------------------------------------------------------------

%-----------------------------------------------------------------------
% Clean Up Workspace
%clear;
%-----------------------------------------------------------------------

return;



function createline(x1,x2,y1,y2,color,width)
%createline - given 2 xcoordinates, 2 ycoordinates, color, width, and line
%name, create a new line in the figure

line([x1 x2],[y1 y2],'Color',color,'LineWidth',width,'Clipping','off');
return;


function createnormal(bxy,branch,bus,gen)
%createnormal    Creates Display of Normal System Topology
%   Dynamically color codes based on bus voltage being too high, or too low
%   or MVA rating of line at either end of line being exceeded
%   Generation and Load is displayed on each bus per the input matrix
%   bxy holds the location of the bus bars (bus x1 x2 y1 y2) and is NX4
%   branch,bus,gen is an output as a result of the power flow solution

%-----------------------------------------------------------------------
% Variables
state = ' Normal'; %current game state

% Sorts and Sums the Generation
gen = sortrows(gen,[1 2])

D = size(gen);
gen2 = gen;
n = 2;
for k=2:D(1)
    if (gen(k,1) == gen(k-1,1))
       gen2(n,2) = gen2(n,2) + gen2(n-1,2);
       gen2(n,3) = gen2(n,3) + gen2(n-1,3);
       gen2(n-1,:) = [];
       n = n-1;
    else
    end;
    n = n+1;
end;

gen = gen2;

% Creates a general data matrx used in this procedure
D = size(bus);
G = size(gen);
busdata = zeros(D(1),7);
for k=1:D(1)
    busdata(k,1) = bus(k,1);
    busdata(k,2) = bus(k,8);
    busdata(k,3) = bus(k,9);
    busdata(k,4) = 0;
    busdata(k,5) = 0;
    for j =1:G(1)
        if bus(k,1) == gen(j,1)
            busdata(k,4) = gen(j,2);
            busdata(k,5) = gen(j,3);
```

```
        else
        end;
    end;
    busdata(k,6) = bus(k,3);
    busdata(k,7) = bus(k,4);
end;

% Creates a Matrix of Bus Labels
D = size(bus);
for k=1:D(1)
bustext(k,1:4) = strcat('Bus',int2str(bus(k,1)));
end;

%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% Set the GRID
cla; % clear Figure Axis
clf; % clear Figure
set(0,'Units','normalized'); % normalizes the screen coordinate system to 0,0,1,1
axis([0 200 0 200])
axis off; %hides the axis
set(gcf,'Color','k') % sets the background of the figure to black
rect = [100,100,600,600];
set(gcf,'Position',rect) % sets the position of the figure
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% Print Current States and Scores
title1 = text(40,210,'Current System Topology','FontSize',17,'Color','w'); % Title
title1 = text(-30,190,strcat('State = ',state),'FontSize',12,'Color','w'); %Game State
title1 = text(-30,180,'Red Values = Over Voltage Limits','FontSize',8,'Color','r'); %info
title1 = text(-30,170,'Red Lines = Over MVA Rating','FontSize',8,'Color','r'); %info
title1 = text(-30,160,'Blue Values = Under Voltage Limit','FontSize',8,'Color','b'); %info
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% Prints Bus Bars and Adds IDs
D=size(bxy);
% prints the bus bars
for k=1:D(1)
createline(bxy(k,2),bxy(k,3),bxy(k,4),bxy(k,5),'w',4);
end;
% prints the bus bar text
for k=1:D(1)
createtext(bustext(k,1:4),bxy(k,2),bxy(k,3),bxy(k,4),bxy(k,5));
end;
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% Create Transmission Lines -  Dynamically color coded based on MVA overload
% Determine the Centroid of the bus bars
[branches] = centroidlines(bxy);
D=size(branch);
for k=1:D(1)
if sqrt(branch(k,12)*branch(k,12) + branch(k,13)*branch(k,13)) > branch(k,6) || sqrt(branch(k,14)*branch(k,14) +
branch(k,15)*branch(k,15)) > branch(k,6)
  createline(branches(branch(k,1),2),branches(branch(k,2),2),branches(branch(k,1),3),branches(branch(k,2),3),'r',1);
else
  createline(branches(branch(k,1),2),branches(branch(k,2),2),branches(branch(k,1),3),branches(branch(k,2),3),'w',1);
end;
end;
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% Print Voltages on Bus Bars - Dynamically Color Codes Values
% Blue = Undervoltage Limit, Red = Overvoltage Limit
D=size(bxy);
busvoltages = zeros(0,0);
for k=1:D(1)
if busdata(k,2) > bus(k,12)
 text([branches(k,2)-5 branches(k,2)-5],[branches(k,3)+4 branches(k,3)+4],num2str(busdata(k,2)),'Color','r');
else
    if busdata(k,2) < bus(k,13)
        text([branches(k,2)-5 branches(k,2)-5],[branches(k,3)+4 branches(k,3)+4],num2str(busdata(k,2)),'Color','b');
    else
        text([branches(k,2)-5 branches(k,2)-5],[branches(k,3)+4 branches(k,3)+4],num2str(busdata(k,2)),'Color','w');
    end;
end;
end;
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% Print Generation and Load
D = size(busdata);
for k=1:D(1)
if busdata(k,4) > 0
text([branches(k,2)-20 branches(k,2)-20],[branches(k,3)-5 branches(k,3)-5],'Generation','Color','y');
text([branches(k,2)-20 branches(k,2)-20],[branches(k,3)-10 branches(k,3)-
10],strcat(num2str(busdata(k,4),4),'W'),'Color','y');
text([branches(k,2)-20 branches(k,2)-20],[branches(k,3)-15 branches(k,3)-
15],strcat(num2str(busdata(k,5),4),'V'),'Color','y');
else
end;
if busdata(k,6) > 0
```

```
text([branches(k,2)+10 branches(k,2)+10],[branches(k,3)-5 branches(k,3)-5],'Load','Color','m');
text([branches(k,2)+10 branches(k,2)+10],[branches(k,3)-10 branches(k,3)-
10],strcat(num2str(busdata(k,6),4),'W'),'Color','m');
text([branches(k,2)+10 branches(k,2)+10],[branches(k,3)-15 branches(k,3)-
15],strcat(num2str(busdata(k,7),4),'V'),'Color','m');
else
end;
end;
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% Clean Up Workspace
%clear;
%--------------------------------------------------------------------------

return;



function [name2] = createtext(name,x1,x2,y1,y2)
%createline - given 2 xcoordinates, 2 ycoordinates, color, width, and line
%name, create a new line in the figure
name2 = strcat(name,'t');
name2 = text([x1+20 x2],[y1 y2],name,'Color','w');
return;


function [br, Sf, St] = ctgcomputebranchflows9bus(bus,gen,branchnew,V,Yf,Yt,baseMVA)
% CTGCOMPUTEBRANCHFLOWS
%   Input: bus data, gen data, branch data,
%   1P1Q final complex V, the matrices Yf and Yt
    % which, when multiplied by a complex voltage vector, yield the vector
    % currents injected into each line from the "from" and "to" buses
    % respectively of each line, baseMVA
%   Outputs: branch numbers in service (br), complex power at the from bus (Sf),
%   complex power at the to bus (St)

%--------------------------------------------------------------------------
% Compute Branch Flows
%--------------------------------------------------------------------------

%%-----  initialize  -----
%% define named indices into bus, gen, branch matrices
[PQ, PV, REF, NONE, BUS_I, BUS_TYPE, PD, QD, GS, BS, BUS_AREA, VM, ...
    VA, BASE_KV, ZONE, VMAX, VMIN, LAM_P, LAM_Q, MU_VMAX, MU_VMIN] = idx_bus;
[F_BUS, T_BUS, BR_R, BR_X, BR_B, RATE_A, RATE_B, ...
    RATE_C, TAP, SHIFT, BR_STATUS, PF, QF, PT, QT, MU_SF, MU_ST] = idx_brch;
[GEN_BUS, PG, QG, QMAX, QMIN, VG, MBASE, ...
    GEN_STATUS, PMAX, PMIN, MU_PMAX, MU_PMIN, MU_QMAX, MU_QMIN] = idx_gen;

% read data & convert to internal bus numbering
[i2e, bus, gen, branchnew] = ext2int(bus, gen, branchnew);


%Branch Flows
br = find(branchnew(:, BR_STATUS));
Sf = V(branchnew(br, F_BUS)) .* conj(Yf(br, :) * V) * baseMVA; % complex power at "from" bus
St = V(branchnew(br, T_BUS)) .* conj(Yt(br, :) * V) * baseMVA; % complex power injected at "to" bus
return;







%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
%========================================================================
% Contingency Case Screening
%========================================================================
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

warning off MATLAB:singularMatrix;          % supresses error messages

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%--------------------------------------------------------------------------
%                                  Base Case
%--------------------------------------------------------------------------
[baseMVA, bus, gen, branch, success] = runpf('wscc9bus');
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%--------------------------------------------------------------------------
%                         Branch Contingency Screening
%--------------------------------------------------------------------------
[bctgCombinedMVAoverloads,bctgCombinedVoltageViolations] = bctgscreening9bus(baseMVA, bus, gen, branch)
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%--------------------------------------------------------------------------
%                         Generator Contingency Screening
%--------------------------------------------------------------------------
```

```
[gctgCombinedMVAoverloads,gctgCombinedVoltageViolations] = gctgscreening9bus(baseMVA, bus, gen, branch)
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<



 function [mc] = gasturbine(N,capacity)

N;               % Economic lifetime (planning period) in years
ahr = 11800;      % Average heate rate BTU/kWh
ic = 350;       % Investment cost $/kW
fc = 3.5;        % $/MBtu
FOM = 1;        % Fixed O&M Cost $/kW/Year
VOM = 5;         % Variable O&M Cost $/MWh
a = 0.08;         % Fuel Cost and O&M Escalation %/year
r = 0.10;        % Discount rate
FCR = 0.18;      % Fixed charge rate
capacity; % capacity in MW
capacityfactor = 0.9;
fprintf('\n   Gas Turbine Plant:  ');
fprintf('\n'   );
fprintf('\n'   );
fprintf('\n   Economic lifetime (planning period) in years:  %s',int2str(N));
fprintf('\n   Average heate rate BTU per kWh:  %s',int2str(ahr));
fprintf('\n   Investment cost $ per kW:  %s',int2str(ic));
fprintf('\n   $/MBtu:  %s',num2str(fc,3));
fprintf('\n   Fixed O&M Cost $ per kW per Year:  %s',num2str(FOM,2));
fprintf('\n   Variable O&M Cost $ per MWh:  %s',int2str(VOM));
fprintf('\n   Fuel Cost and O&M Escalation percent per year:  %s',num2str(a,2));
fprintf('\n   Discount Rate:  %s',num2str(r,2));
fprintf('\n   Fixed Charge Rate:  %s',num2str(FCR,2));
fprintf('\n   Capacity in MW:  %s',int2str(capacity));
fprintf('\n   Capacity Factor:  %s',num2str(capacityfactor,2));

% Capital Recovery Factor
CRF = r*((1+r)^N)/((1+r)^N-1);
fprintf('\n   Capital Recovery Factor:  %s',num2str(CRF,6));
% Levelizing Factor for uniform inflation
LF = CRF*[1-(((1+a)/(1+r))^N)]/(r-a);
fprintf('\n   Levelizing Factor for uniform inflation:  %s',num2str(LF,6));
% Fixed Levelized Annual Costs
investment = ic*FCR*1000;                % Investment $/MW/YEAR
fixedom = FOM*LF*1000;                   % Fixed O&M $/MW/YEAR
k1 = investment + fixedom;               % Total $/MW/YEAR
fprintf('\n   Fixed Levelized Annual Costs $ per MW per YEAR:  %s',num2str(k1,7));
% Variable Levelized Annual Costs
fuel = ahr*fc*LF/1000;                   % Fuel $/MWh
VOMc = VOM*LF;                           % Var. O&M $/MWH
c1 = fuel + VOMc;                        % Total $/MWH
fprintf('\n   Variable Levelized Annual Costs $ per MWH :  %s',num2str(c1,4));
% Total Investment Cost
tic = (capacity*ic*10^3)/10^6;  % $ Millions
fprintf('\n   Total Investment Cost $ Millions:  %s',num2str(tic,3));
% Operating Costs
% Yearly Operating Cost
c1a = ((ahr*fc*1000)/10^6) +VOM;   % $/MWH
fprintf('\n   Yearly Operating Cost $ per MWH :  %s',num2str(c1,3));
% Per Year
yoc = capacity*capacityfactor*8760*c1a/10^6;   % $ M/yr
fprintf('\n   Yearly Operating Cost $ M per yr:  %s',num2str(yoc,3));
fprintf('\n');

mc = [k1 c1]; % Returns fixed and variable levelized annual costs
return;
function [MVAoverloads] = gctgcalculateMVAoverloads9bus(genout,branchnew,Sf,St)
%GCTGCALCULATEMVAOVERLOADS - Calculates MVA violations for generator
% outage
%
% Inputs: 1P1Q estimated voltage V, bus data, from bus, to bus
% Outputs: Condensed matrix with ONLY violated MVA
% violations
%
% gctgCombinedMVAoverloads: (NX4 where N is the number of violations
% [<-generator contingency --->  <---branch overloaded-->  <---violation--->
% [     generator                 from_bus         to_bus   percent_overload]


%-------------------------------------------------------------------------
% Compute MVA Overloads
%-------------------------------------------------------------------------

MVAoverloads = zeros(8,4);

for i=1:8
    if abs(Sf(i,1)) > branchnew(i,6)
        MVAoverloads(i,1) = branchnew(i,1);
        MVAoverloads(i,2) = branchnew(i,2);
        MVAoverloads(i,3) = 100*abs(Sf(i,1))./branchnew(i,6);
    else
    end;
    if abs(St(i,1)) > branchnew(i,6)
        MVAoverloads(i,1) = branchnew(i,1);
        MVAoverloads(i,2) = branchnew(i,2);
        MVAoverloads(i,4) = 100*abs(St(i,1))./branchnew(i,6);
    else
```

```
    end
end;

%Eliminate zero rows
n=8;
while n >0
    if MVAoverloads(n,1) == 0
        MVAoverloads(n,:) = [];
    else
    end
    n = n-1;
end;

% Determines the Max overload between the from and to end of the branch

D = size(MVAoverloads);
MVAoverloads2 = zeros(D(1),3);
for i =1:D(1)
    MVAoverloads2(i,1) = MVAoverloads(i,1);
    MVAoverloads2(i,2) = MVAoverloads(i,2);
    if MVAoverloads(i,3) > MVAoverloads(i,4)
        MVAoverloads2(i,3) = MVAoverloads(i,3);
    else
        MVAoverloads2(i,3) = MVAoverloads(i,4);
    end
end;

MVAoverloads = MVAoverloads2;
D = size(MVAoverloads);
MVAoverloads2 = zeros(D(1),4);
for i=1:D(1)
    MVAoverloads2(i,1) = genout;
    MVAoverloads2(i,2) = MVAoverloads(i,1);
    MVAoverloads2(i,3) = MVAoverloads(i,2);
    MVAoverloads2(i,4) = MVAoverloads(i,3);
end;
MVAoverloads = MVAoverloads2;
return;

function [VoltageViolations] = gctgcalculateVoltageViolations9bus(V,bus,genout)
%GCTGCALCULATEVOLTAGEVIOLATIONS - Calculates voltage violations for
%generator outage
%
% Inputs: 1P1Q estimated voltage V, bus data, from bus, to bus
% Outputs: Condensed matrix with ONLY violated bus voltage
% violations
%
% gctgCombinedVoltageViolations: (NX8) where N is the number of violations
% [<---branch contingency --->  <---bus violated--><-----------------under voltage---------------><--------over
voltage--------------------------->
% [generator                          bus             current_value low_limit percent_undervoltage  current_value
high_limit  percent_overvoltage]

Vm = abs(V);
%--------------------------------------------------------------------------
% Compute Buses Overvoltage
%--------------------------------------------------------------------------
VoltageViolations = zeros(9,7);

% Undervoltage Violations
for i=1:9
    if Vm(i,1) < bus(i,13)
            VoltageViolations(i,1) = i;
            VoltageViolations(i,2) = Vm(i,1);
            VoltageViolations(i,3) = bus(i,13);
            VoltageViolations(i,4) = Vm(i,1)./bus(i,13);
    else
    end;
end;

%OverVoltage Violations
for i=1:9
    if Vm(i,1) > bus(i,12)
      VoltageViolations(i,1) = i;
      VoltageViolations(i,5) = Vm(i,1);
      VoltageViolations(i,6) = bus(i,12);
      VoltageViolations(i,7) = Vm(i,1)./bus(i,12);
    else
    end;
end;

% Because MATLAB math is not so correct this is needed to correct it at limits
for i=1:9
    if abs(Vm(i,1) - bus(i,12)) < 0.00000005
      VoltageViolations(i,1) = 0;
    else
    end;
end;

%Eliminate zero rows
n=9;
while n >0
    if VoltageViolations(n,1) == 0
        VoltageViolations(n,:) = [];
    else
```

```
    end;
    n = n-1;
end;

VoltageViolations;

D = size(VoltageViolations);
VoltageViolations2 = zeros(D(1),8);
for i=1:D(1)
    VoltageViolations2(i,1) = genout;
    VoltageViolations2(i,2) = VoltageViolations(i,1);
    VoltageViolations2(i,3) = VoltageViolations(i,2);
    VoltageViolations2(i,4) = VoltageViolations(i,3);
    VoltageViolations2(i,5) = VoltageViolations(i,4);
    VoltageViolations2(i,6) = VoltageViolations(i,5);
    VoltageViolations2(i,7) = VoltageViolations(i,6);
    VoltageViolations2(i,8) = VoltageViolations(i,7);
end;

VoltageViolations = VoltageViolations2;
clear VoltageViolations2;
return;



function plotloadforecast(year,loadforecast)

D = size(year);

subplot(3,3,1)
plot(year,loadforecast(1,1:D(2)),'-mo','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',[.49 1
.63],'MarkerSize',6);
title('Annual Load Forecast Bus 1');
xlabel('Year');
ylabel('MW Demand');

subplot(3,3,2)
plot(year,loadforecast(2,1:D(2)),'-mo','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',[.49 1
.63],'MarkerSize',6);
title('Annual Load Forecast Bus 2');
xlabel('Year');
ylabel('MW Demand');

subplot(3,3,3)
plot(year,loadforecast(3,1:D(2)),'-mo','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',[.49 1
.63],'MarkerSize',6);
title('Annual Load Forecast Bus 3');
xlabel('Year');
ylabel('MW Demand');

subplot(3,3,4)
plot(year,loadforecast(4,1:D(2)),'-mo','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',[.49 1
.63],'MarkerSize',6);
title('Annual Load Forecast Bus 4');
xlabel('Year');
ylabel('MW Demand');

subplot(3,3,5)
plot(year,loadforecast(5,1:D(2)),'-mo','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',[.49 1
.63],'MarkerSize',6);
title('Annual Load Forecast Bus 5');
xlabel('Year');
ylabel('MW Demand');

subplot(3,3,6)
plot(year,loadforecast(6,1:D(2)),'-mo','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',[.49 1
.63],'MarkerSize',6);
title('Annual Load Forecast Bus 6');
xlabel('Year');
ylabel('MW Demand');

subplot(3,3,7)
plot(year,loadforecast(7,1:D(2)),'-mo','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',[.49 1
.63],'MarkerSize',6);
title('Annual Load Forecast Bus 7');
xlabel('Year');
ylabel('MW Demand');

subplot(3,3,8)
plot(year,loadforecast(8,1:D(2)),'-mo','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',[.49 1
.63],'MarkerSize',6);
title('Annual Load Forecast Bus 8');
xlabel('Year');
ylabel('MW Demand');

subplot(3,3,9)
plot(year,loadforecast(9,1:D(2)),'-mo','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',[.49 1
.63],'MarkerSize',6);
title('Annual Load Forecast Bus 9');
xlabel('Year');
ylabel('MW Demand');

return;
```

```
 function
printctg(bctgCombinedMVAoverloads,bctgCombinedVoltageViolations,gctgCombinedMVAoverloads,gctgCombinedVoltageViolations)

fprintf('\n');
fprintf('\n');
fprintf('\n_____________________________________________________________________________');
fprintf('\n');
fprintf('\n               N-1 Branch Contingency Results                        ');
fprintf('\n');
fprintf('\n');
fprintf('\n Screening routine to examine each non-islanding branch outage in the 9 bus system. ');
fprintf('\n Estimates all post-contingency voltage magnitudes and all');
fprintf('\n post-contingency MVA branch flows.');
fprintf('\n')
fprintf('\n Inputs: Base case AC power flow solutions with baseMVA, bus, gen, and');
fprintf('\n branch matrices for the 9 bus system');
fprintf('\n Outputs: Condensed matrix with ONLY violated branch MVA overloads and bus voltage');
fprintf('\n violations');
fprintf('\n');
fprintf('\n');
fprintf('\n');
fprintf('\n');
fprintf('\n MVA Overloads Due to Any Branch Outage');
fprintf('\n');
fprintf('\n bctgCombinedMVAoverloads: (NX5) where N is the number of violations');
fprintf('\n [<---branch contingency --->  <---branch overloaded-->  <---violation--->');
fprintf('\n [from_bus            to_bus  from_bus         to_bus   percent_overload]');
fprintf('\n');
fprintf('\n');
bctgCombinedMVAoverloads
fprintf('\n');
fprintf('\n');
fprintf('\n Voltage Violations Due to Any Branch Outage');
fprintf('\n');
fprintf('\n bctgCombinedVoltageViolations: (NX9) where N is the number of violations');
fprintf('\n');
fprintf('\n [<---branch contingency --->  <---bus violated--><-----------------under voltage--------------><--------
over voltage--------------------------> ');
fprintf('\n [from_bus            to_bus       bus             current_value low_limit percent_undervoltage
current_value   high_limit  percent_overvoltage]');
fprintf('\n');
fprintf('\n');
bctgCombinedVoltageViolations


fprintf('\n');
fprintf('\n');
fprintf('\n_____________________________________________________________________________');
fprintf('\n');
fprintf('\n               N-1 Generator Contingency Results                       ');
fprintf('\n');
fprintf('\n');
fprintf('\n Screening routine to examine each generator unit outage (except the swing) in the  ');
fprintf('\n 9 bus system. Estimates all post-contingency voltage magnitudes');
fprintf('\n and all post-contingency MVA branch flows.');
fprintf('\n')
fprintf('\n Inputs: Base case AC power flow solutions with baseMVA, bus, gen, and');
fprintf('\n branch matrices for the 9 bus system');
fprintf('\n Outputs: Condensed matrix with ONLY violated branch MVA overloads and bus voltage');
fprintf('\n violations');
fprintf('\n');
fprintf('\n');
fprintf('\n');
fprintf('\n');



fprintf('\n MVA Overloads Due to Any Generator Outage');
fprintf('\n');
fprintf('\n gctgCombinedMVAoverloads: (NX4 where N is the number of violations');
fprintf('\n [<-generator contingency --->  <---branch overloaded-->  <---violation--->');
fprintf('\n [     generator bus            from_bus         to_bus   percent_overload]');
fprintf('\n');
fprintf('\n');
gctgCombinedMVAoverloads
fprintf('\n');
fprintf('\n');
fprintf('\n Voltage Violations Due to Any Generator Outage');
fprintf('\n');
fprintf('\n gctgCombinedVoltageViolations: (NX8) where N is the number of violations');
fprintf('\n');
fprintf('\n [<---generator contingency --->  <---bus violated--><-----------------under voltage--------------><-------
-over voltage-------------------------->  ');
fprintf('\n [       generator bus                    bus      current_value low_limit percent_undervoltage
current_value   high_limit  percent_overvoltage]');
fprintf('\n');
fprintf('\n');
gctgCombinedVoltageViolations



return;
```

```
function [MVAoverloads] = bctgcalculateMVAoverloads9bus(branchnew,Sf,St,f,t)
%BCTGCALCULATEMVAOVERLOADS - Calculates MVA violations for branch
% outages
%
% Inputs: 1P1Q estimated voltage V, bus data, from bus, to bus
% Outputs: Condensed matrix with ONLY violated MVA
% violations
%
% bctgCombinedMVAoverloads: (NX5) where N is the number of violations
% [<---branch contingency --->  <---branch overloaded-->  <---violation--->
% [from_bus             to_bus  from_bus         to_bus   percent_overload]


%--------------------------------------------------------------------------
% Compute MVA Overloads
%--------------------------------------------------------------------------
D = size(branchnew);
MVAoverloads = zeros(D(1),4);

for i=1:D(1)
    if abs(Sf(i,1)) > branchnew(i,6)
        MVAoverloads(i,1) = branchnew(i,1);
        MVAoverloads(i,2) = branchnew(i,2);
        MVAoverloads(i,3) = 100*abs(Sf(i,1))./branchnew(i,6);
    else
    end;
    if abs(St(i,1)) > branchnew(i,6)
        MVAoverloads(i,1) = branchnew(i,1);
        MVAoverloads(i,2) = branchnew(i,2);
        MVAoverloads(i,4) = 100*abs(St(i,1))./branchnew(i,6);
    else
    end
end;

%Eliminate zero rows
n=D(1);
while n >0
    if MVAoverloads(n,1) == 0
        MVAoverloads(n,:) = [];
    else
    end
    n = n-1;
end;

% Determines the Max overload between the from and to end of the branch

D = size(MVAoverloads);
MVAoverloads2 = zeros(D(1),3);
for i =1:D(1)
    MVAoverloads2(i,1) = MVAoverloads(i,1);
    MVAoverloads2(i,2) = MVAoverloads(i,2);
    if MVAoverloads(i,3) > MVAoverloads(i,4)
        MVAoverloads2(i,3) = MVAoverloads(i,3);
    else
        MVAoverloads2(i,3) = MVAoverloads(i,4);
    end
end;

MVAoverloads = MVAoverloads2;
D = size(MVAoverloads);
MVAoverloads2 = zeros(D(1),5);
for i=1:D(1)
    MVAoverloads2(i,1) = f;
    MVAoverloads2(i,2) = t;
    MVAoverloads2(i,3) = MVAoverloads(i,1);
    MVAoverloads2(i,4) = MVAoverloads(i,2);
    MVAoverloads2(i,5) = MVAoverloads(i,3);
end;
MVAoverloads = MVAoverloads2;
clear D;
return;



function [MVAbase, bus, gen, branch, success, et] = ...
                bctgrunpf(branchnew, casename, mpopt, fname, solvedcase)
%BCTGRUNPF  Runs a power flow specifically for branch contingency.
%
%   [baseMVA, bus, gen, branch, success, et] = ...
%           runpf(casename, mpopt, fname, solvedcase)
%
%   Runs a power flow (full AC Newton's method by default) and optionally
%   returns the solved values in the data matrices, a flag which is true if
%   the algorithm was successful in finding a solution, and the elapsed time
%   in seconds. All input arguments are optional. If casename is provided it
%   specifies the name of the input data file or struct (see also 'help
%   caseformat' and 'help loadcase') containing the power flow data. The
%   default value is 'case9'. If the mpopt is provided it overrides the
%   default MATPOWER options vector and can be used to specify the solution
%   algorithm and output options among other things (see 'help mpoption' for
%   details). If the 3rd argument is given the pretty printed output will be
%   appended to the file whose name is given in fname. If solvedcase is
%   specified the solved case will be written to a case file in MATPOWER
%   format with the specified name. If solvedcase ends with '.mat' it saves
%   the case as a MAT-file otherwise it saves it as an M-file.
```

```
%
%   If the ENFORCE_Q_LIMS options is set to true (default is false) then if
%   any generator reactive power limit is violated after running the AC power
%   flow, the corresponding bus is converted to a PQ bus, with Qg at the
%   limit, and the case is re-run. The voltage magnitude at the bus will
%   deviate from the specified value in order to satisfy the reactive power
%   limit. If the reference bus is converted to PQ, the first remaining PV
%   bus will be used as the slack bus for the next iteration. This may
%   result in the real power output at this generator being slightly off
%   from the specified values.

%   MATPOWER
%   $Id: runpf.m,v 1.10 2005/01/18 22:48:32 ray Exp $
%   by Ray Zimmerman, PSERC Cornell
%   Enforcing of generator Q limits inspired by contributions
%   from Mu Lin, Lincoln University, New Zealand (1/14/05).
%   Copyright (c) 1996-2005 by Power System Engineering Research Center (PSERC)
%   See http://www.pserc.cornell.edu/matpower/ for more info.
warning off MATLAB:singularMatrix;          % supresses error messages
%%-----  initialize  -----
%% define named indices into bus, gen, branch matrices
[PQ, PV, REF, NONE, BUS_I, BUS_TYPE, PD, QD, GS, BS, BUS_AREA, VM, ...
    VA, BASE_KV, ZONE, VMAX, VMIN, LAM_P, LAM_Q, MU_VMAX, MU_VMIN] = idx_bus;
[F_BUS, T_BUS, BR_R, BR_X, BR_B, RATE_A, RATE_B, ...
    RATE_C, TAP, SHIFT, BR_STATUS, PF, QF, PT, QT, MU_SF, MU_ST] = idx_brch;
[GEN_BUS, PG, QG, QMAX, QMIN, VG, MBASE, ...
    GEN_STATUS, PMAX, PMIN, MU_PMAX, MU_PMIN, MU_QMAX, MU_QMIN] = idx_gen;

%% default arguments
if nargin < 4
    solvedcase = '';                %% don't save solved case
    if nargin < 3
        fname = '';                 %% don't print results to a file
        if nargin < 2
            mpopt = mpoption;       %% use default options
            if nargin < 1
                casename = 'case9'; %% default data file is 'case9.m'
            end
        end
    end
end

%% options
verbose = mpopt(31);
qlim = mpopt(6);                    %% enforce Q limits on gens?
dc = mpopt(10);                     %% use DC formulation?

%% read data & convert to internal bus numbering
[baseMVA, bus, gen, branch] = loadcase(casename);

branch = branchnew;
[i2e, bus, gen, branch] = ext2int(bus, gen, branch);

%% get bus index lists of each type of bus
[ref, pv, pq] = bustypes(bus, gen);

%% generator info
on = find(gen(:, GEN_STATUS) > 0);      %% which generators are on?
gbus = gen(on, GEN_BUS);                %% what buses are they at?

%%-----  run the power flow  -----
t0 = clock;
if dc                               %% DC formulation
    %% initial state
    Va0 = bus(:, VA) * (pi/180);

    %% build B matrices and phase shift injections
    [B, Bf, Pbusinj, Pfinj] = makeBdc(baseMVA, bus, branch);

    %% compute complex bus power injections (generation - load)
    %% adjusted for phase shifters and real shunts
    Pbus = real(makeSbus(baseMVA, bus, gen)) - Pbusinj - bus(:, GS) / baseMVA;

    %% "run" the power flow
    Va = dcpf(B, Pbus, Va0, ref, pv, pq);

    %% update data matrices with solution
    branch(:, [QF, QT]) = zeros(size(branch, 1), 2);
    branch(:, PF) = (Bf * Va + Pfinj) * baseMVA;
    branch(:, PT) = -branch(:, PF);
    bus(:, VM) = ones(size(bus, 1), 1);
    bus(:, VA) = Va * (180/pi);
    %% update Pg for swing generator (note: other gens at ref bus are accounted for in Pbus)
    %%      Pg = Pinj + Pload + Gs
    %%      newPg = oldPg + newPinj - oldPinj
    refgen = find(gbus == ref);             %% which is(are) the reference gen(s)?
    gen(on(refgen(1)), PG) = gen(on(refgen(1)), PG) + (B(ref, :) * Va - Pbus(ref)) * baseMVA;

    success = 1;
else                                %% AC formulation
    %% initial state
    % V0    = ones(size(bus, 1), 1);            %% flat start
    V0  = bus(:, VM) .* exp(sqrt(-1) * pi/180 * bus(:, VA));
    V0(gbus) = gen(on, VG) ./ abs(V0(gbus)).* V0(gbus);
```

```
    if qlim
        ref0 = ref;                         %% save index and angle of
        Varef0 = bus(ref0, VA);             %%   original reference bus
        limited = [];                       %% list of indices of gens @ Q lims
        fixedQg = zeros(size(gen, 1), 1);   %% Qg of gens at Q limits
    end
    repeat = 1;
    while (repeat)
        %% build admittance matrices
        [Ybus, Yf, Yt] = makeYbus(baseMVA, bus, branch);

        %% compute complex bus power injections (generation - load)
        Sbus = makeSbus(baseMVA, bus, gen);

        %% run the power flow
        alg = mpopt(1);
        if alg == 1
            [V, success, iterations] = newtonpf(Ybus, Sbus, V0, ref, pv, pq, mpopt);
        elseif alg == 2 | alg == 3
            [Bp, Bpp] = makeB(baseMVA, bus, branch, alg);
            [V, success, iterations] = fdpf(Ybus, Sbus, V0, Bp, Bpp, ref, pv, pq, mpopt);
        elseif alg == 4
            [V, success, iterations] = gausspf(Ybus, Sbus, V0, ref, pv, pq, mpopt);
        else
            error('Only Newton''s method, fast-decoupled, and Gauss-Seidel power flow algorithms currently
implemented.');
        end

        %% update data matrices with solution
        [bus, gen, branch] = pfsoln(baseMVA, bus, gen, branch, Ybus, Yf, Yt, V, ref, pv, pq);

        if qlim             %% enforce generator Q limits
            %% find gens with violated Q constraints
            mx = find( gen(:, GEN_STATUS) > 0 & gen(:, QG) > gen(:, QMAX) );
            mn = find( gen(:, GEN_STATUS) > 0 & gen(:, QG) < gen(:, QMIN) );

            if ~isempty(mx) | ~isempty(mn)  %% we have some Q limit violations
                if verbose & ~isempty(mx)
                    fprintf('Gen %d at upper Q limit, converting to PQ bus\n', mx);
                end
                if verbose & ~isempty(mn)
                    fprintf('Gen %d at lower Q limit, converting to PQ bus\n', mn);
                end

                %% save corresponding limit values
                fixedQg(mx) = gen(mx, QMAX);
                fixedQg(mn) = gen(mn, QMIN);
                mx = [mx;mn];

                %% convert to PQ bus
                gen(mx, QG) = fixedQg(mx);      %% set Qg to binding limit
                gen(mx, GEN_STATUS) = 0;        %% temporarily turn off gen,
                for i = 1:length(mx)            %% (one at a time, since
                    bi = gen(mx(i), GEN_BUS);   %%  they may be at same bus)
                    bus(bi, [PD,QD]) = ...      %% adjust load accordingly,
                        bus(bi, [PD,QD]) - gen(mx(i), [PG,QG]);
                end
                bus(gen(mx, GEN_BUS), BUS_TYPE) = PQ;   %% & set bus type to PQ

                %% update bus index lists of each type of bus
                ref_temp = ref;
                [ref, pv, pq] = bustypes(bus, gen);
                if verbose & ref ~= ref_temp
                    fprintf('Bus %d is new slack bus\n', ref);
                end
                limited = [limited; mx];
            else
                repeat = 0; %% no more generator Q limits violated
            end
        else
            repeat = 0;     %% don't enforce generator Q limits, once is enough
        end
    end
    if qlim & ~isempty(limited)
        %% restore injections from limited gens (those at Q limits)
        gen(limited, QG) = fixedQg(limited);    %% restore Qg value,
        for i = 1:length(limited)               %% (one at a time, since
            bi = gen(limited(i), GEN_BUS);      %%  they may be at same bus)
            bus(bi, [PD,QD]) = ...              %% re-adjust load,
                bus(bi, [PD,QD]) + gen(limited(i), [PG,QG]);
        end
        gen(limited, GEN_STATUS) = 1;               %% and turn gen back on
        if ref ~= ref0
            %% adjust voltage angles to make original ref bus correct
            bus(:, VA) = bus(:, VA) - bus(ref0, VA) + Varef0;
        end
    end
end
et = etime(clock, t0);

%%-----  output results  -----
%% convert back to original bus numbering & print results
[bus, gen, branch] = int2ext(i2e, bus, gen, branch);

%% this is just to prevent it from printing baseMVA
```

```
%% when called with no output arguments
if nargout, MVAbase = baseMVA; end

return;



function [MVAbase, bus, gen, gencost, branch, f, success, et] = ...
               runuopfspecialnoprint(baseMVA, bus, gen, branch, areas, gencost,mpopt)
%RUNUOPF  Runs an optimal power flow with unit-decommitment heuristic.
%
%   [baseMVA, bus, gen, gencost, branch, f, success, et] = ...
%           runuopf(casename, mpopt, fname, solvedcase)
%
%   Runs an optimal power flow with a heuristic which allows it to shut down
%   "expensive" generators and optionally returns the solved values in the
%   data matrices, the objective function value, a flag which is true if the
%   algorithm was successful in finding a solution, and the elapsed time in
%   seconds. All input arguments are optional. If casename is provided it
%   specifies the name of the input data file or struct (see also 'help
%   caseformat' and 'help loadcase') containing the opf data. The default
%   value is 'case9'. If the mpopt is provided it overrides the default
%   MATPOWER options vector and can be used to specify the solution
%   algorithm and output options among other things (see 'help mpoption' for
%   details). If the 3rd argument is given the pretty printed output will be
%   appended to the file whose name is given in fname. If solvedcase is
%   specified the solved case will be written to a case file in MATPOWER
%   format with the specified name. If solvedcase ends with '.mat' it saves
%   the case as a MAT-file otherwise it saves it as an M-file.

%   MATPOWER
%   $Id: runuopf.m,v 1.7 2004/08/23 20:59:38 ray Exp $
%   by Ray Zimmerman, PSERC Cornell
%   Copyright (c) 1996-2004 by Power System Engineering Research Center (PSERC)
%   See http://www.pserc.cornell.edu/matpower/ for more info.

%%-----  initialize  -----
%% default arguments

%% read data & convert to internal bus numbering
[i2e, bus, gen, branch, areas] = ext2int(bus, gen, branch, areas);

%% run unit commitment / optimal power flow
[bus, gen, branch, f, success, et] = uopf(baseMVA, bus, gen, gencost, branch, areas, mpopt);

%% convert back to original bus numbering & print results
[bus, gen, branch, areas] = int2ext(i2e, bus, gen, branch, areas);



%% this is just to prevent it from printing baseMVA
%% when called with no output arguments
if nargout, MVAbase = baseMVA; end

return;


 function [MVAbase, bus, gen, gencost, branch, f, success, et] = ...
               runuopfspecial(baseMVA, bus, gen, branch, areas, gencost,mpopt)
%RUNUOPF  Runs an optimal power flow with unit-decommitment heuristic.
%
%   [baseMVA, bus, gen, gencost, branch, f, success, et] = ...
%           runuopf(casename, mpopt, fname, solvedcase)
%
%   Runs an optimal power flow with a heuristic which allows it to shut down
%   "expensive" generators and optionally returns the solved values in the
%   data matrices, the objective function value, a flag which is true if the
%   algorithm was successful in finding a solution, and the elapsed time in
%   seconds. All input arguments are optional. If casename is provided it
%   specifies the name of the input data file or struct (see also 'help
%   caseformat' and 'help loadcase') containing the opf data. The default
%   value is 'case9'. If the mpopt is provided it overrides the default
%   MATPOWER options vector and can be used to specify the solution
%   algorithm and output options among other things (see 'help mpoption' for
%   details). If the 3rd argument is given the pretty printed output will be
%   appended to the file whose name is given in fname. If solvedcase is
%   specified the solved case will be written to a case file in MATPOWER
%   format with the specified name. If solvedcase ends with '.mat' it saves
%   the case as a MAT-file otherwise it saves it as an M-file.

%   MATPOWER
%   $Id: runuopf.m,v 1.7 2004/08/23 20:59:38 ray Exp $
%   by Ray Zimmerman, PSERC Cornell
%   Copyright (c) 1996-2004 by Power System Engineering Research Center (PSERC)
%   See http://www.pserc.cornell.edu/matpower/ for more info.

%%-----  initialize  -----
%% default arguments

%% read data & convert to internal bus numbering
[i2e, bus, gen, branch, areas] = ext2int(bus, gen, branch, areas);

%% run unit commitment / optimal power flow
[bus, gen, branch, f, success, et] = uopf(baseMVA, bus, gen, gencost, branch, areas, mpopt);
```

```
%% convert back to original bus numbering & print results
[bus, gen, branch, areas] = int2ext(i2e, bus, gen, branch, areas);

printpf(baseMVA, bus, gen, branch, f, success, et, 1, mpopt);


%% this is just to prevent it from printing baseMVA
%% when called with no output arguments
if nargout, MVAbase = baseMVA; end

return;
```