ECE 4320 Fall 2020 Course Project: Power System Pandemic

Course Instructor: Daniel Molzahn, Georgia Tech ECE

1 Introduction

This project places you in charge of operating, maintaining, and repairing a simulated model of the Georgia power system during a pandemic. Your goal is to provide a low-cost supply of power to the loads by carefully choosing the generator outputs and allocating crews to perform maintenance and repairs of the transmission lines. Sending crews out to maintain and repair transmission lines may result in them being quarantined after exposure to the pandemic disease, precluding their use from the remainder of the simulation. You must therefore balance the need to maintain low-cost and reliable power system operation against the health risk imposed on the repair and maintenance crews. The remainder of this document describes the simulation and your task for the project.

2 Simulation

2.1 Dependencies and Installation

The simulation code and all data files are given in Matlab format in `ece4320project.zip`. The simulation code is developed for Matlab 2019a. To avoid potential incompatibilities with Matlab, you should install version 2019a or later. Note that you will need the Mapping toolbox in Matlab, which is available with the Georgia Tech Matlab license.

The simulation relies on MATPOWER (`https://matpower.org`), an open-source Matlab package that is included in this zip file. MATPOWER solves power flow and optimal power flow problems. While we will discuss MATPOWER and its associated data format in more detail during lecture, you are encouraged to install and play with this tool on your own. The MATPOWER manual provides an extensive description of the functionality of this package: `https://matpower.org/docs/MATPOWER-manual.pdf`

The simulation additionally uses another open-source Matlab package called YALMIP, which is also included in the `ece4320project.zip` file. YALMIP is a so-called "modeling" package that simplifies the formulation and solution of optimization problems. This project uses YALMIP to formulate power flow and security-constrained unit commitment problems that you can use as tools for determining generator setpoints and load shedding amounts during the simulation.

You can install the entire simulation (including both MATPOWER and YALMIP) by simply unzipping `ece4320project.zip` and adding the associated files and folders to your Matlab path.

You can do this either by using the "Set Path" button on the Matlab toolbar or by using the following command:

```
addpath(genpath('c:\path\to\unzipped\files'));
```

The simulation has a number of associated parameters that are stored in the file `setParameters.m`. These parameters can be directly edited by changing the values of the variables stored in this file.

2.2 Dataset

This project makes use of a fictitious dataset representing the Georgia power system with 20 buses, 6 generators, and 30 transmission lines. This dataset is available in MATPOWER format in the file `case20ga.m`. MATPOWER datasets are structured according to the format given in Section 3 and Appendix B of the MATPOWER manual. You can also use MATPOWER's `define_constants` command to define indexing variables that allow straightforward access to various quantities in the MATPOWER data format.

While the power system dataset is geographically placed in Georgia and the load demands are roughly proportional to the actual load demands, this system is completely fictitious and does not match the actual power grid model. Nevertheless, the behavior of this system is fairly typical.

Using the dataset provided in the file `case20ga.m`, the simulation considers various snapshots in time, with the number of time periods determined by the parameter `lastTurn`. For the sake of simplicity, each period of the simulation is assumed to correspond to the same amount of elapsed time. At the start of the simulation, the dataset for each time period is stored into the Matlab structure `mpcs`, where `mpcs{i}` holds the MATPOWER dataset associated with time period `i`.[1]

2.3 Power System Model

2.3.1 Power Flow Model

The power system behavior is modeled using the DC power flow approximation. Hence, only active power (generation, load demands, and line flows) and voltage angles are considered in the formulation. Reactive power (generation, load demands, and line flows) and voltage magnitudes are neglected. The relevant parameters for the lines are the reactances, given in the dataset (in per unit) by `mpcs{i}.branch(:,BR_X)`,[2] and the flow limits (in MW) by `mpcs{i}.branch(:,RATE_A)`.

2.3.2 Time-Varying Load Demands

Each time period has different load demands. For time period `i`, the load demands for each bus are stored in `mpcs{i}.bus(:,PD)`.

The time periods cycle between representing night, day, and evening time periods. The selected time periods are specified via the parameter `opts.TimesOfDay`. The evening periods generally have the highest load demands, while the night periods have the lowest load demands. The simulation starts on Monday of the week indicated by the parameter `opts.StartWeek` (numeric index with week 1 representing the first week in January and 52 indicating the last week in December).

[1] The variable name `mpcs` corresponds to MATPOWER cases.

[2] Note that the variable `BR_X` is obtained by calling `define_constants`. Other indexing variables for the quantities stored in `mpcs` are also obtained by calling `define_constants`.

In addition to the daily and weekly cycles, the load demands at each bus are multiplicatively modulated by Gaussian random variables characterized by the parameters `opts.Pmu` (mean value) and `opts.Psigma` (standard deviation). Larger values of `opts.Pmu` increase the load demands. Larger values of `opts.Psigma` increase the amount of variation in the load demands.

The load demands stored in `mpcs{i}.bus(:,PD)` indicate the maximum load that can be supplied at each bus. In some instances, you may want to shed load (i.e., deliver less than the load demands specified in `mpcs{i}.bus(:,PD)`) in order to prevent line overloads. Each MWh of load shed incurs the cost specified in `mpcs{i}.loadcost`, given in dollars per MWh. Note that the cost of load shedding is the same for each load and each time period.

2.3.3 Generator Model

The bus number of each generator is given in `mpcs{i}.gen(:,GEN_BUS)`. The generation cost is modeled as an affine function of the active power output ($c_1 P_G + c_0$), with linear coefficients c_1 given in `mpcs{i}.gencost(:,COST+1)` and constant terms c_0 given in `mpcs{i}.gencost(:,COST+2)`. The cost cofficients stored in the dataset are given in dollars per MWh of power generated. The order of the rows for the generation costs in `mpcs{i}.gencost` matches the order of the rows for the generators in `mpcs{i}.gen`.

As implied by the affine cost function, the no-load cost for each generator (the cost for having the generator turned on during this time period) is given by the constant term c_0 in `mpcs{i}.gencost(:,COST+2)`. If the generator is turned on, the power output P_G must be between the upper and lower active power generation limits `mpcs{i}.gen(:,PMAX)` and `mpcs{i}.gen(:,PMIN)` (in MW) and the associated cost is $c_1 P_G + c_0$. Generators that are turned off, as indicated by an active power output equal to zero, do not incur any generation cost.

Note that there is no coupling between the time periods for the generator quantities. In other words, the simulation does not consider issues such as generator ramp rates, startup and shutdown time requirements, etc. Note also that the generator cost coefficients are the same for each time period.

2.3.4 Line Failure Model

At the start of the simulation, all lines are in-service (not failed) and thus have a line status equal to 1 (`mpcs{i}.branch(:,BR_STATUS) = 1`). Throughout the simulation, lines may fail either exogenously or due to cascading failures when line flows are greater than the corresponding line flow limits, resulting in the line status becoming less than 1. All lines with status less than 1 are unavailable until repaired. Note that the simulation does not consider the possibility of generator failures.

Exogenous Line Failures At time period `i`, each line has a exogenous probability of failure given in `mpcs{i}.failure_probability`. This failure probability models events outside the simulation, such as severe storms. The exogenous failure probabilities can be reduced by assigning crews to perform preventative maintenance, which will be discussed later in this section.

The exogenous failure probabilities for most lines are randomly initialized between the values given in the parameters `opts.minFailureProbability` and `opts.failProb`. A fraction `opts.outlierFraction` of lines are initialized with a larger exogenous failure probability that

is approximately `opts.outlierFailProb`. This is used to model lines which are due for preventative maintenance. Starting from these initializations, the exogenous failure probabilities for each line increase during each time period by a multiplicative factor of `opts.FailureRateIncrease`.

At the beginning of each time period, the simulation randomly determines which lines fail according to the lines' exogenous failure probabilities in that time period. The statuses for these lines are changed to 0 (`mpcs{i}.branch(:,BR_STATUS) = 0`) and these lines thus become unavailable until they are fully repaired.

Cascading Failures After determining the exogeneous failures, the simulation then evaluates a cascading failure model based on the power flowing on each line after these failures occur. Lines which are overloaded (have power flows greater than the flow limits specified in `mpcs{i}.branch(:,RATE_A)`) may fail. Overloaded lines fail with a probability that varies linearly with the amount of overload. The failure probability for an overloaded line is 0% when the line flow is at its limit and 100% when the line is over the limit by more than a factor of `opts.instaFail`.

When a line fails due to overloading, the line's status is set to `opts.cascadeFailureStatus`. While this can be modified, the default value for this parameter in `setParameters.m` is set so that the line can be repaired by one crew in a single turn. This is due to the fact that line overloads generally result in the operation of the protection system (opening a breaker), which is relatively straightforward to correct compared to the physical damage that is modeled by exogenous failures.

The cascading failure model considers the possibility of multiple rounds of failures. Each round introduces random failures among the overloaded lines, recomputes the line flows with the newly failed lines out of service, and then determines whether any additional lines fail due to overloads from the new line flows. This process repeats until there are no further line failures during this turn.

2.3.5 Maintenance Model

During each turn, you can allocate crews that either perform maintenance on lines which have not yet failed or repair failed lines. Crews that are allocated to in-service (non-failed) lines will perform maintenance on those lines. Each crew assigned to a line reduces the line's exogeneous failure probability by a multiplicative factor of `opts.maintenanceEffectiveness` for all subsequent turns in the simulation. Note that the exogeneous failure probability can never become less than `opts.minFailureProbability` regardless of the amount of maintenance performed on the line.

Note that reductions in the exogeneous failure probabilities occur after computing the line failures for the turn. In other words, maintenance improves the failure probabilities for subsequent time periods but not the current time period. If a line fails during a turn (either exogeneously or after the cascading failure analysis), any crews assigned to this line will perform repairs rather than maintenance.

2.3.6 Repair Model

Crews that are allocated to failed lines perform repairs on that line. Each crew assigned to a failed line increases the line's status by an amount specified by `opts.repairEffectiveness`. Once a failed line's status would become greater than or equal to 1, the line's status is set to 1 and the

line returns to service. All lines whose statuses are less than 1 are considered to be failed and cannot carry any power flow (i.e., they are entirely removed from the power flow model).

Note that repairs occur after computing the line failures for the turn. In other words, repairs are applied to subsequent time periods but not the current time period. If a line fails during a turn (either exogeneously or after the cascading failure analysis), any crews assigned to this line will perform repairs rather than maintenance.

2.4 Quarantine Model

At the start of the simulation, you are allocated a number of repair crews equal to `ncrew` for performing maintenance and repair activities. Crews sent into the field for these activities may be exposed to the pandemic disease and thus quarantined. Quarantined crews are unavailable for repair and maintenance activities for the remainder of the simulation.

Each crew that is allocated during a turn is randomly exposed to the disease and quarantined with a probability given in `opts.proximitySicknessLikelihood`. Crews that are not sent into the field (unallocated) have no risk of being quarantined during that turn. If multiple crews are performing maintenance or repairs on a single line, any crew becoming exposed results in the quarantining of all crews allocated to that line. You will therefore need to balance the value in rapidly performing maintenance or repairing a failed line via allocating multiple crews to that line against the likelihood of quarantining multiple crews.

In addition to losing the crews' capabilities for the remainder of the simulation, quarantining crews also significantly impacts the crews' quality of life. To model these impacts, the final score associated with a simulation is penalized by a multiplicative factor of `opts.crewQuarantinePenalty` for each quarantined crew at the end of the simulation.

2.5 Scoring Methodology

The final score for the simulation is determined by three components:

1. Load shedding costs as determined by the cost coefficient `mpcs{i}.loadcost` (Section 2.3.2).
2. Generation costs as determined by the cost coefficients in `mpcs{i}.gencost(:,COST+1)` and `mpcs{i}.gencost(:,COST+2)` (Section 2.3.3).
3. A multiplicative penalty based on the number of crews quarantined at the end of the simulation and the parameter `opts.crewQuarantinePenalty` (Section 2.4.)

The generation and load shedding costs are summed over the time periods and then multiplied by the quarantine penalty factor at the end of the simulation.

2.6 Computational Tools

In order to manage the generator setpoints and the load shedding, you are given two computational tools in the file `ece4320project.zip` that are similar to those available to actual power system operators. Specifically, you are given solvers for DC power flow problems and DC security-constrained unit commitment problems. As part of your work on the project, you are welcome to use, adapt, and build on these functions to determine better operating points and crew allocations.

2.6.1 DC Power Flow

The first function is `[sol]=rundcpf_with_islands(mpc,Pd,Pg)` ("Run DC Power Flow With Islands"). This function solves a DC power flow problem, i.e., given the loads and generator outputs, this function computes the power flows on the lines.

The input `mpc` is a single period of the power system model in MATPOWER format. Failed lines (status values that are less than 1, `mpc.branch(:,BR_STATUS) < 1`) are automatically removed from the model prior to solving the problem.

The inputs `Pd` and `Pg` are vectors of the load demands and the generator outputs (both in MW). Note that these may differ from the values for these quantities stored in the input `mpc`. Differences between the load demands in `mpc.bus(:,PD)` and `Pd` indicate the amount of load shedding.

The function `rundcpf_with_islands` builds on the MATPOWER function `rundcpf`, which also solves DC power flow problems. However, the MATPOWER function `rundcpf` does not handle power systems with islands (cases where the transmission network has multiple disconnected components), whereas the function `rundcpf_with_islands` does handle islands.[3]

Example function call:

```
% Run a DC power flow with 50% load shedding at each bus
% and equal output from each generator.

define_constants;
ngen = size(mpcs{currentTurn}.gen,1);   % number of generators

Pd = mpcs{currentTurn}.bus(:,PD) * 0.5;
Pg = sum(Pd) / ngen;
[sol]=rundcpf_with_islands(mpcs{currentTurn},Pd,Pg);
printpf(sol); % Show the solution
```

2.6.2 DC Security-Constrained Unit Commitment

The second function is `[sol]=rundcscuc(mpc,contingencies)` ("Run DC Security-Constrained Unit Commitment"). This function solves DC security-constrained unit commitment problems specified via two inputs: `mpc`, a power system dataset in MATPOWER format, and `contingencies`, a matrix indicating a set of contingencies.

The input `mpc` must correspond to a single time period. Failed lines (status values that are less than 1, `mpc.branch(:,BR_STATUS) < 1`) are automatically removed from the model prior to solving the problem.

The input `contingencies` is a matrix with rows corresponding to the transmission lines and columns corresponding to the contingencies that you want to consider in the security-constrained unit commitment problem. Each column denotes a contingency scenario, with values of 1 in the

[3] If the power system data in the input `mpc` has islands, the function `rundcpf_with_islands` uniformly sheds load or reduces generation within each island in order to achieve power balance (total load equal to total generation within the island) and then runs a DC power flow in each island.

entries of that column indicating the failures of the corresponding transmission lines. For instance, a column with values equal to 1 in the second and fourth entry corresponds to a contingency consisting of the failures of the second and fourth transmission lines.

The output `sol` provides the generator setpoints and load demands that minimize the generation costs while avoiding constraint violations in the base case as well as remaining secure against all of the specified contingency scenarios. Note that the function `rundcscuc` may turn off generators (power output set to zero) and shed loads (values of `sol.bus(:,PD)` that are less than `mpc.bus(:,PD)`).

Similar to `rundcpf_with_islands`, the function `rundcscuc` automatically handles cases where the power system is composed of multiple islands.

Example function call:

```
% Run a DC security-constrained unit commitment.
% Consider a system with 30 lines and 3 contingencies.
% The three contingencies are:
%   The outage of line 1.
%   The outages of lines 2 and 4.
%   The outages of lines 1, 2, and 5.

define_constants;

contingencies = zeros(30,3);      % 30 lines and 3 contingencies
contingencies(1,1) = 1;           % contingency 1
contingencies([2, 4], 2) = 1;     % contingency 2
contingencies([1, 2, 5], 3) = 1; % contingency 3

[sol]=rundcscuc(mpcs{currentTurn},contingencies);
printpf(sol); % Show the solution
```

2.7 User Interface

You can interface with the simulation using either GUI-based commands (in combination with the Matlab command line) or by defining a function that provides the crew allocations, generator setpoints, and load shedding using the information available from each turn of the simulation. Each of these options are described in detail below.

2.7.1 GUI-Based Interface

The first method for interfacing with the simulation uses GUI-based commands and the Matlab command line. The GUI in Figure 1 allows you to perform several functions:

- **Visualize Next Turn Button**: Changes the display to the current turn, reads the values for active power generation, active power load, and crew allocation in the variables `Pg`, `Pd`, and `crewAllocation`, respectively, that are stored in the Matlab workspace, and then

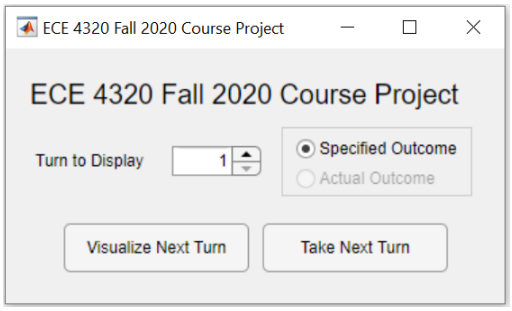

Figure 1: GUI for Controlling the Simulation and Changing the Display.

displays the results. Note that this is the "specified outcome" that does not simulate line failures or crew quarantines. This can be therefore be viewed as the best-case scenario based on your specified values of `Pg`, `Pd`, and `crewAllocation`.

- **Take Next Turn Button**: Takes the next turn (including simulating line failures and crew quarantines) based on the values for active power generation, active power load, and crew allocation in the variables `Pg`, `Pd`, and `crewAllocation`, respectively, that are stored in the Matlab workspace, and then displays the results. This is the "actual outcome" for this turn.

- **Turn to Display Selector**: Changes the displayed values to those associated with a particular turn. Selecting the current turn shows the specified outcome (using the values of `Pg`, `Pd`, and `crewAllocation` stored in the Matlab workspace). Selecting a future turn shows the loads and flows assuming zero output from every generator. This is useful for seeing the loads that will occur during future turns. Selecting a prior turn shows either specified outcome or the actual outcome from that turn, depending on the Outcome indicator.

- **Outcome Indicator**: If the turn to display is prior to the current turn, this indicator selects the display of either the specified outcome (without simulating line failures and crew quarantines) or the actual outcome (including any line failures and crew quarantines that occurred).

Thus, you can interact with the simulation by first storing appropriate values for the power generation, load demands, and crew allocations in the variables `Pg`, `Pd`, and `crewAllocation` via the Matlab command line and then clicking the Take Next Turn Button. You can visualize the impacts of prospective setpoints using the Visualize Next Turn Button and see the specified and actual outcomes of past turns using the Turn to Display Selector and Outcome Indicator.

Each turn, the simulation updates the following variables in the Matlab workspace:

1. `mpcs`: The variable containing the power system model (network topology, impedances, generator limits, load demands, failed lines, etc. in the MATPOWER format as well as the line failure probabilities) for each time period as described in Section 2.3.

2. `ncrew`: The number of crews available at this turn.

3. `currentTurn`: The current turn number in the simulation. The expression `mpcs{currentTurn}` corresponds to the system model during the current turn in the simulation.

4. `lastTurn`: The final turn number in the simulation.

The values stored in the variables `Pg`, `Pd`, and `crewAllocation` must conform to the following formats in order to take the next turn:

1. `Pg`: A column vector with length equal to the number of generators and values indicating the power outputs of each generator for the current turn. All generator outputs must be either zero, indicating that the generator is shut down, or within the corresponding generator limits (`mpcs{currentTurn}.gen(:,PMAX)` and `mpcs{currentTurn}.gen(:,PMIN)`).

2. `Pd`: A column vector with length equal to the number of buses indicating the load demands at each bus for the current turn. All load demands must be between zero and the values of the demands specified in `mpcs{currentTurn}.bus(:,PD)`. Values of `Pd` that are less than the power demands in `mpcs{currentTurn}.bus(:,PD)` correspond to load shedding.

3. `crewAllocation`: A column vector of length `ncrew` which indicates how your available crews are allocated to maintain and repair the lines during the current turn. The entries must be either zero, indicating that this crew is not allocated, or correspond to a valid line number.

The total power generation must equal the total power demand (`sum(Pg) == sum(Pd)`). At the start of each turn, these vectors are initialized with zero values.

The GUI also includes a one-line diagram of the power system, as shown in Figure 2 and the zoomed display in Figure 3. The one-line diagram indicates the buses via black dots and the transmission lines via the black lines. Each bus is labeled with the bus name and the bus number is shown in brackets. Each bus also shows the quantity of fulfilled power demand P_d and the total power demand at that bus. (These values are equal if there is no load shedding.) If a bus has a generator, then the power generation P_g is shown along with the upper and lower limits for the output of that generator. Each line is labeled with a line number corresponding to the row of this line in `mpcs{i}.branch`. For instance, the second line corresponds to `mpcs{i}.branch(2,:)`. Arrows show the direction of power flow on each line, and the quantities of the power flowing on each line are given both in MW and as a percentage value in terms of the line's flow limit (i.e., as a fraction of the values in `mpcs{i}.branch(:,RATE_A)`). Lines that have failed (out of service) are shown in red along with the line status (i.e., a fractional value indicating how much repair work is required to restore the line).[4] The display also shows the number of crews assigned to each line to perform maintenance or repair work (as appropriate given the line's status).

The typical Matlab figure tools allow you to zoom in, zoom out, and scroll on the one-line diagram. In recent versions of Matlab, these tools can be accessed by hovering your cursor over the figure and then clicking the icons that appear in the upper right corner.

2.7.2 Function-Based Interface

As an alternative to the GUI interface, you can interact with the simulation via user-defined functions. This is useful for automatically performing many simulations to characterize the effectiveness of a particular strategy without having to point and click through the GUI interface.

[4]Recall from Section 2.3.4 that each line has a status value stored in `mpcs{i}.branch(:,BR_STATUS)`. Lines with status less than 1 are failed (out of service). These lines can be repaired by assigning crews to them, with each crew increasing a line's status by the amount specified in `opts.repairEffectiveness` per turn.

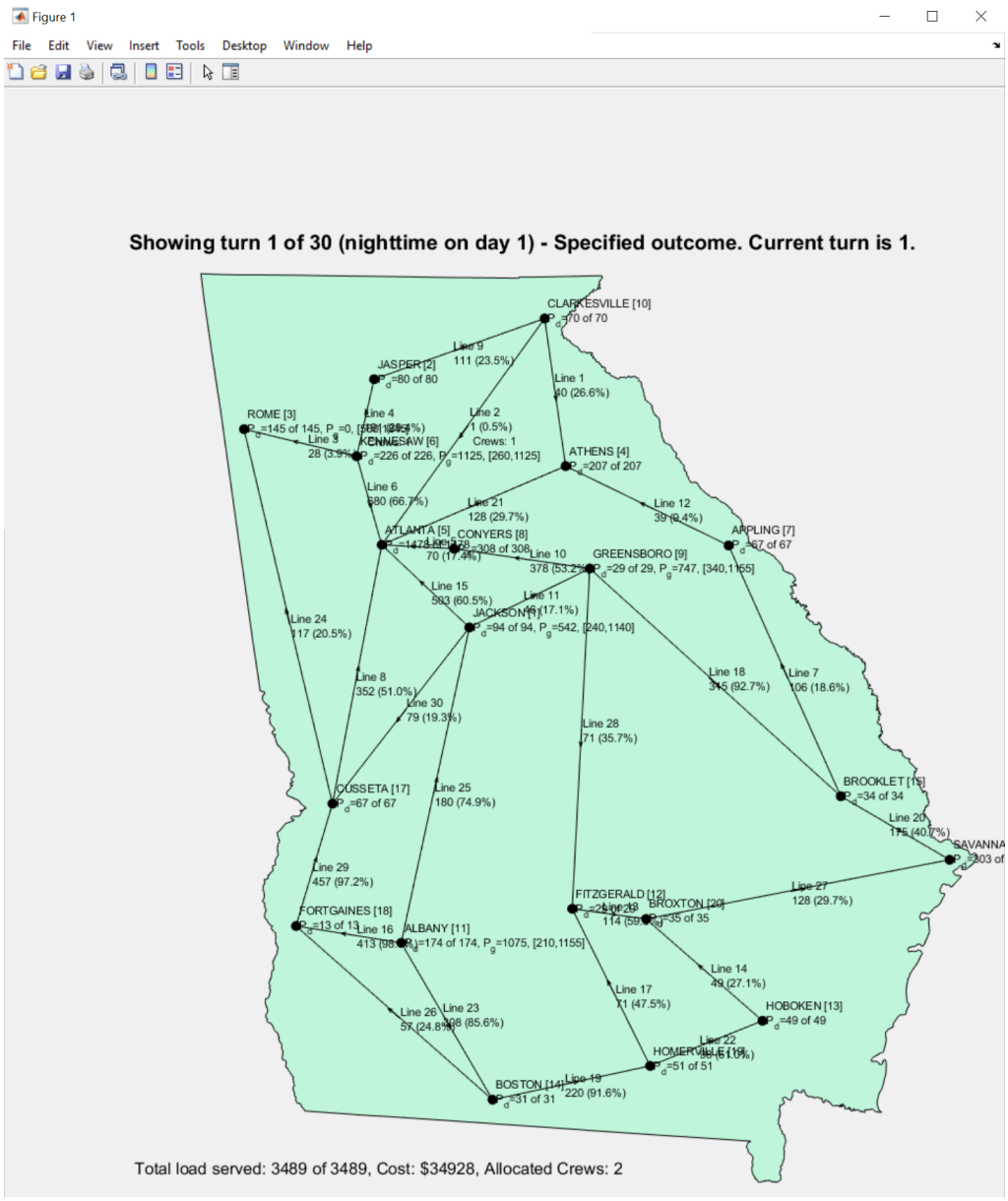

Figure 2: GUI Visualizing the Simulation.

This function-based interface with the simulation is performed using the function

`[mpcs,Pd,mpcs_specified,Pd_specified,crewAllocation,ncrew,score]`
`= power_system_pandemic(function_name,display_results)`.

The first input to the function `power_system_pandemic`, `function_name`, is a string containing the name of a user-defined function that takes the inputs `mpcs`, `ncrew`, `currentTurn`, `lastTurn` and provides the outputs `Pg`, `Pd`, `crewAllocation`, in this order.

These inputs and outputs are defined in Section 2.7.1 and correspond to the information available to the user and expected by the simulation at each turn. In other words, you provide a function (with named specified by the input `function_name` to `power_system_pandemic`) that

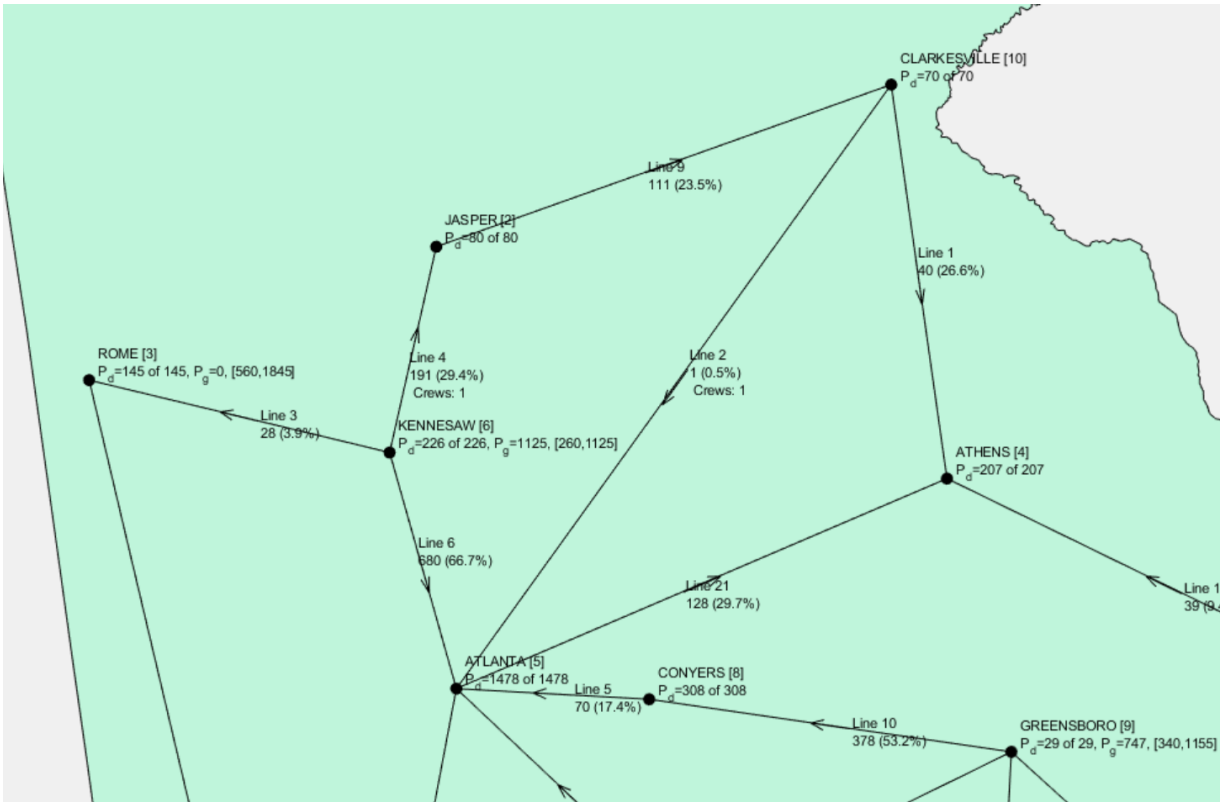

Figure 3: GUI Visualizing the Simulation (Zoomed).

performs whatever computations you want in order to obtain `Pg`, `Pd`, and `crewAllocation` for each turn. Thus, any strategy that you can encode programmatically can be quickly tested by calling `power_system_pandemic`. By calling `power_system_pandemic` multiple times, you can quickly evaluate the performance of your strategy over many simulations.

The second input to `power_system_pandemic`, `display_results`, is a Boolean flag indicating whether the GUI should be displayed with the result of the simulation after the code terminates.

The outputs of `power_system_pandemic` give the final results of the simulation, including all inputs specified (in `mpcs_specified`, `Pd_specified`, `crewAllocation`), the outputs produced (in `mpcs`, `Pd`, `ncrew`) throughout the simulation, and the final score (in `score`).[5]

3 Project Tasks and Deliverables

You are free to choose one of the two following tasks to complete for this project. This choice will not affect your grade in the sense that high-quality work on the first task will be scored the same as high-quality work on the second task. The first task asks you to make recommendations for responding to a range of possible pandemic scenarios, while the second asks for a detailed algorithm for one particular pandemic scenario.

[5]Note that the power demands in `Pd` might not always match those specified in `Pd_specified` if load must be automatically shed to maintain power balance when cascading line failures result in an islanded power system.

3.1 Task 1: Parameter Sensitivity Analysis

If you choose this first task, you will take the role of a utility engineer asked to develop an operating procedure report that will be used during future pandemics. This report will provide instructions for operating, maintaining, and repairing the system during a range of pandemic scenarios. Your goal is to provide the reader of this report with the knowledge needed to operate the system reliably and inexpensively despite the challenges associated with future pandemics.

To develop this report, you will need to run multiple simulations in order to account for the random aspects of the simulation related to the load demands, line failures, and crew quarantines. Moreover, future pandemics might have a range of parameters (for instance, different levels of contagiousness, time of occurrence, and duration) and the lines may have differing levels of failure probabilities, repair speed, and maintenance effectiveness due to the amount of prior maintenance. You will therefore need to also perform multiple simulations while varying the simulation parameters. (As a reminder, you can edit these parameters by changing the file `setParameters.m`.) In other words, your report should provide a *parameter sensitivity analysis* for all parameters that you think are relevant.

When developing your recommendations, you may want to consider questions such as:

- Which security constraints should be enforced (no contingencies, all $N-1$ contingencies, some $N-k$ contingencies for $k>1$, etc.)?
- Which values should be imposed for the line flow limits (exactly the values specified in `mpcs{currentTurn}.branch(:,RATE_A)` or some smaller or larger values)?
- How to allocate the crews to repair and maintain the lines?
- Should your approach change as the simulation proceeds?

You have two deliverables for this task:

1. A status update describing your progress on the project, due at the end of the day on November 16, 2020.
2. A written final project report, due at the end of the final exam period (December 2, 2020 at 5:30 pm eastern time). Note: there is no final exam for this course.

Your status update should be a one-page or less document describing the work you have done to date on your project. In this status update, you should also discuss the challenges or difficulties that you have encountered and your plans for addressing these challenges in order to successfully complete the project and final report by the deadline. This status update is worth 5% of your project grade.

Your final project report should include:

- an appropriate title for the report,
- an executive summary that briefly summarizes the purpose of the report and provides your key recommendations,
- an introduction discussing the impacts of pandemics on electric power grids,

- a summary of the pandemic simulation model described in Section 2 of this document,
- a description of your experimental setup indicating how you selected the parameters in the simulation,
- the results of your analysis with recommendations for operating, maintaining, and repairing the system as the parameters vary,
- any limitations or caveats associated with your analysis,
- a conclusion summarizing your report.

You can assume that the reader of your report has a basic knowledge of power systems operation and therefore understands terms like "DC power flow", "security constraints", and "unit commitment", but is not familiar with this simulation tool itself.

You can use any format for this report. You should follow professional writing practices, e.g., cite the sources you use for external information (any standard citation format is acceptable, but be consistent in the citation format for all references), write clearly and concisely, organize your report in an easy to follow manner with section headings, thoroughly proofread to eliminate typos and grammatical errors, etc. You are encouraged to use clear and appropriate figures (graphs, tables, decision diagrams, etc.) in your report to illustrate your technical approach and results.

3.2 Task 2: Operational Algorithm

If you choose this second task, you will take the role of a utility engineer responsible for responding to an ongoing pandemic. In this task, you can assume that you know the parameters associated with this particular pandemic (i.e., all the parameters in the file `setParameters.m` are fixed). Your goal is to determine how to best operate, maintain, and repair the system during this pandemic by developing an algorithm that computes the generator outputs, load shedding, and crew allocations. This algorithm should interface with the simulation via the "function-based interface" described in Section 2.7.2.

Your goal is to develop an algorithm that performs well, as judged by the score at the last turn, for many simulations conducted using the same parameters. When developing your algorithm, you may want to consider questions such as:

- Which security constraints should be enforced (no contingencies, all $N-1$ contingencies, some $N-k$ contingencies for $k>1$, etc.)?
- Which values should be imposed for the line flow limits (exactly the values specified in `mpcs{currentTurn}.branch(:,RATE_A)` or some smaller or larger values)?
- How to allocate the crews to repair and maintain the lines?
- Should your approach change as the simulation proceeds?

You are welcome to use any tools that you would like to develop and analyze your algorithm. For instance, you could run many simulations to do statistical analyses of the performance achieved by various algorithms or to train machine learning tools.

You have three deliverables for this task:

1. A status update describing your progress on the project, due at the end of the day on November 16, 2020.

2. The code implementing your algorithm, due at the end of the final exam period (December 2, 2020 at 5:30 pm eastern time). Note: there is no final exam for this course.

3. A written final project report, due at the end of the final exam period.

Your status update should be a one-page or less document describing the work you have done to date on your project. In this status update, you should also discuss the challenges or difficulties that you have encountered and your plans for addressing these challenges in order to successfully complete the project and final report by the deadline. This status update is worth 5% of your project grade.

Your code should follow good programming practices, including adequate documentation with comments in order to be readable by other programmers. Your code should run when called by the function `power_system_pandemic` as described in Section 2.7.2.

Your final report should include:

- an appropriate title for the report,
- an executive summary that briefly summarizes your algorithm,
- an introduction discussing the impacts of pandemics on electric power grids,
- a summary of the pandemic simulation model described in Section 2 of this document,
- a detailed description of your algorithm,
- a discussion of your experimental setup indicating how your analyzed your algorithm,
- an analysis of your algorithm with comparisons to alternative approaches that you considered,
- a conclusion summarizing your report.

You can assume that the reader of your report has a basic knowledge of power systems operation and therefore understands terms like "DC power flow", "security constraints", and "unit commitment", but is not familiar with this simulation tool itself.

You can use any format for this report. You should follow professional writing practices, e.g., cite the sources you use for external information (any standard citation format is acceptable, but be consistent in the citation format for all references), write clearly and concisely, organize your report in an easy to follow manner with section headings, thoroughly proofread to eliminate typos and grammatical errors, etc. You are encouraged to use clear and appropriate figures (graphs, tables, decision diagrams, etc.) in your report to illustrate your technical approach and results.

3.3 Deadlines

Regardless of the task you select for your project, your project deliverables are due by the following deadlines:

1. Project Status Report: End of the day on November 16, 2020.
2. Written Final Project Report (both tasks) and code (Task 2): End of the final exam period (December 2, 2020 at 5:30 pm eastern time). There is no final exam for ECE 4320.

4 Grading Rubric

For either of the tasks, your deliverables will be graded according to the following rubric.

	Exceeds Expectations	Meets Expectations	Does Not Meet Expectations
Overall Technical Contribution (60%)			
Technical Approach (30%)	Appropriate and reasonable approach for solving the technical problem.	Overall technical approach is reasonable but has some minor flaws or shortcomings.	Approach has serious flaws or does not solve the technical problem.
Quality of Results (20%)	Capabilities and advantages/disadvantages of the technical approach are successfully and convincingly demonstrated.	The technical approach is demonstrated successfully, but has some minor inadequacies or shortcoming.	Demonstrations of the technical approach are nonexistent or severely lacking in appropriateness.
Creativity (10%)	Several aspects of the technical approach are performed with a high degree of creativity.	The technical approach has some moderately creative aspects.	The technical approach lacks creativity.
Written Report (35%)			
Structure (5%)	All paragraphs are well-organized. Section layout is logical.	Paragraphs are usually well-organized. Section layout is logical.	Paragraphs are poorly organized. Section layout is illogical.
Content (15%)	Technical discussions are clear and comprehensive. Figures enhance the readability of the report.	Technical discussions are mostly clear. Figures generally support the project results.	Technical discussions lack clarity.
Writing Quality (10%)	At most a few minor grammatical issues or word choice errors.	Sentences are generally well-written. There are a few incorrect word choices or grammatical errors that do not overly distract from the report's legibility.	Sentences are poorly written. There are numerous incorrect word choices and grammatical errors. The report's legibility is significantly flawed.
References (5%)	Detailed references and discussion explain the technical content in context of other work. Citations use a consistent format.	References adequately provide context for the project work. Citations are generally consistent.	The report fails to correctly or adequately reference related work.
Project Status Update (5%)			
Status Update (5%)		Updated provided by November 16th.	

Appendix

GUI-Based Example

This appendix first provides an illustrative example of interacting with the simulation via the GUI.

```
% Start the simulation with the GUI
ece4320;

% Define constants to help reference quantities in the Matpower format.
define_constants;

% Display the lines with the highest failure probability
sortrows([(1:30).' mpcs{currentTurn}.failure_probability],2,'descend')

ans =
   28.0000    0.0658
   25.0000    0.0469
   24.0000    0.0101
   18.0000    0.0097
   27.0000    0.0094
    2.0000    0.0078
   26.0000    0.0075
    7.0000    0.0073
    8.0000    0.0069
   30.0000    0.0062
   21.0000    0.0061
    9.0000    0.0056
   12.0000    0.0055
   23.0000    0.0049
   19.0000    0.0048
   17.0000    0.0047
   14.0000    0.0044
   10.0000    0.0042
   11.0000    0.0040
    1.0000    0.0039
   20.0000    0.0037
    3.0000    0.0036
   22.0000    0.0036
   15.0000    0.0035
   29.0000    0.0034
   16.0000    0.0032
    6.0000    0.0024
   13.0000    0.0023
    5.0000    0.0022
    4.0000    0.0021

% Assign crews to maintain the top two lines in terms of failure probability.
% Two crews assigned to the first, one crew assigned to the second.
crewAllocation(1) = 28; % Assign the first crew to line 28
crewAllocation(2) = 28; % Assign the second crew to line 28
crewAllocation(3) = 25; % Assign the third crew to line 25

% Run a security-constrained unit commitment to determine the generator
% outputs and any load shedding.
contingencies = eye(30,30); % Consider all single-line failure contingencies.
[sol] = rundcscuc(mpcs{currentTurn},contingencies);
printpf(sol); % Show the solution
```

```
Converged in 0.00 seconds
Objective Function Value = 23853.56 $/hr
================================================================================
|     System Summary                                                           |
================================================================================

How many?                How much?              P (MW)            Q (MVAr)
---------------------    -------------------  -------------  -----------------
Buses              20     Total Gen Capacity    7635.0           0.0 to 0.0
Generators          6     On-line Capacity      7635.0           0.0 to 0.0
Committed Gens      6     Generation (actual)   3491.1               0.0
Loads              20     Load                  3491.1               0.0
  Fixed            20       Fixed               3491.1               0.0
  Dispatchable      0       Dispatchable          -0.0 of -0.0      -0.0
Shunts              0     Shunt (inj)           -0.0                 0.0
Branches           30     Losses (I^2 * Z)       0.00                0.00
Transformers        0     Branch Charging (inj)    -                 0.0
Inter-ties          0     Total Inter-tie Flow    0.0                0.0
Areas               1

                          Minimum                      Maximum
                    ------------------------  --------------------------------
Voltage Magnitude   1.000 p.u. @ bus 1          1.000 p.u. @ bus 1
Voltage Angle     -13.60 deg   @ bus 4         20.61 deg   @ bus 11
Lambda P            0.00 $/MWh @ bus 1          0.00 $/MWh @ bus 1
Lambda Q            0.00 $/MWh @ bus 1          0.00 $/MWh @ bus 1

================================================================================
|     Bus Data                                                                 |
================================================================================
 Bus      Voltage          Generation             Load          Lambda($/MVA-hr)
  #   Mag(pu) Ang(deg)   P (MW)   Q (MVAr)   P (MW)   Q (MVAr)     P        Q
----- ------- --------  --------  --------  --------  --------  -------  -------
    1  1.000    9.193    456.18      0.00     94.20      0.00     0.000      -
    2  1.000   -8.191       -         -       80.40      0.00     0.000      -
    3  1.000   -1.019      0.00      0.00    145.10      0.00     0.000      -
    4  1.000  -13.600       -         -      206.90      0.00     0.000      -
    5  1.000  -13.372       -         -     1477.50      0.00     0.000      -
    6  1.000    0.000*  1125.00      0.00    225.70      0.00     0.000      -
    7  1.000  -10.478       -         -       66.80      0.00     0.000      -
    8  1.000   -8.765       -         -      308.30      0.00     0.000      -
    9  1.000   14.321    783.25      0.00     28.60      0.00     0.000      -
   10  1.000  -12.914       -         -       71.30      0.00     0.000      -
   11  1.000   20.613    666.67      0.00    174.20      0.00     0.000      -
   12  1.000   10.668       -         -       29.10      0.00     0.000      -
   13  1.000    5.826       -         -       48.80      0.00     0.000      -
   14  1.000   10.105       -         -       31.30      0.00     0.000      -
   15  1.000    4.404       -         -       34.00      0.00     0.000      -
   16  1.000   11.438    460.00      0.00    302.60      0.00     0.000      -
   17  1.000    2.667       -         -       67.20      0.00     0.000      -
   18  1.000    5.514       -         -       13.30      0.00     0.000      -
   19  1.000    9.210       -         -       50.90      0.00     0.000      -
   20  1.000    8.203       -         -       34.90      0.00     0.000      -
                         --------  --------  --------  --------
               Total:    3491.10      0.00   3491.10      0.00

================================================================================
|     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     10      4     12.16      0.00    -12.16      0.00     0.000      0.00
   2      5     10    -14.25      0.00     14.25      0.00     0.000      0.00
   3      3      6    -79.46      0.00     79.46      0.00     0.000      0.00
   4      6      2    178.11      0.00   -178.11      0.00     0.000      0.00
   5      5      8   -121.80      0.00    121.80      0.00     0.000      0.00
```

```
   6       6       5     641.73       0.00    -641.73       0.00      0.000       0.00
   7       7      15    -248.43       0.00     248.43       0.00      0.000       0.00
   8       5      17    -277.03       0.00     277.03       0.00      0.000       0.00
   9      10       2     -97.71       0.00      97.71       0.00      0.000       0.00
  10       8       9    -430.10       0.00     430.10       0.00      0.000       0.00
  11       1       9     -76.51       0.00      76.51       0.00      0.000       0.00
  12       7       4     181.63       0.00    -181.63       0.00      0.000       0.00
  13      20      12     -16.30       0.00      16.30       0.00      0.000       0.00
  14      20      13      28.96       0.00     -28.96       0.00      0.000       0.00
  15       1       5     435.80       0.00    -435.80       0.00      0.000       0.00
  16      11      18     252.49       0.00    -252.49       0.00      0.000       0.00
  17      12      19      30.06       0.00     -30.06       0.00      0.000       0.00
  18       9      15     172.59       0.00    -172.59       0.00      0.000       0.00
  19      14      19      40.68       0.00     -40.68       0.00      0.000       0.00
  20      16      15     109.83       0.00    -109.83       0.00      0.000       0.00
  21       4       5     -13.11       0.00      13.11       0.00      0.000       0.00
  22      19      13      19.84       0.00     -19.84       0.00      0.000       0.00
  23      14      11    -151.26       0.00     151.26       0.00      0.000       0.00
  24      17       3      65.64       0.00     -65.64       0.00      0.000       0.00
  25       1      11     -88.72       0.00      88.72       0.00      0.000       0.00
  26      14      18      79.29       0.00     -79.29       0.00      0.000       0.00
  27      20      16     -47.57       0.00      47.57       0.00      0.000       0.00
  28       9      12      75.46       0.00     -75.46       0.00      0.000       0.00
  29      17      18    -318.47       0.00     318.47       0.00      0.000       0.00
  30       1      17      91.40       0.00     -91.40       0.00      0.000       0.00
                                                                    --------    --------
                                                            Total:     0.000       0.00

================================================================================
|     Generation Constraints                                                   |
================================================================================
 Gen   Bus                  Active Power Limits
  #     #     Pmin mu     Pmin        Pg        Pmax     Pmax mu
----  -----   -------   --------  --------  --------  -------
   2     3     0.000    560.00       -     1845.00      -
   3     6       -      260.00   1125.00   1125.00     0.000
   6    16     0.000    460.00    460.00   1215.00      -


% Show any load shedding
sortrows([(1:20).' mpcs{currentTurn}.bus(:,PD)-sol.bus(:,PD)],2,'descend')

ans =
    2.0000    0.0000
   19.0000    0.0000
    1.0000         0
    3.0000         0
    4.0000         0
    5.0000         0
    6.0000         0
    7.0000         0
    8.0000         0
    9.0000         0
   10.0000         0
   11.0000         0
   13.0000         0
   14.0000         0
   15.0000         0
   16.0000         0
   17.0000         0
   18.0000         0
   20.0000         0
   12.0000   -0.0000

% Observe that the solution does not have any load shedding (all values equal to zero above).


```

```
% Assign the solution from the security-constrained unit commitment to the appropriate variables
Pg = sol.gen(:,PG);
Pd = sol.bus(:,PD);

% Clicking the Take Next Turn button in the GUI results in the following output.
% This output corresponds to what actually happened on the simulation for this turn.
% At the bottom of this output, we see that there is no load shedding, so the total
% cost is just the generation cost. We also see that the line maintenance has reduced
% the failure probabilities at the lines where we allocated our crews earlier.
% More generally, we would also see updates on line failures, line repairs, and crews
% that are quarantined, when applicable.
Update for Turn 1:
Converged in 0.00 seconds
Objective Function Value = 23853.56 $/hr
================================================================================
|     System Summary                                                           |
================================================================================

How many?                How much?              P (MW)            Q (MVAr)
---------------------    -------------------  -------------  -----------------
Buses              20     Total Gen Capacity    7635.0           0.0 to 0.0
Generators          6     On-line Capacity      7635.0           0.0 to 0.0
Committed Gens      6     Generation (actual)   3491.1               0.0
Loads              20     Load                  3491.1               0.0
  Fixed            20       Fixed               3491.1               0.0
  Dispatchable      0       Dispatchable          -0.0 of -0.0      -0.0
Shunts              0     Shunt (inj)             -0.0               0.0
Branches           30     Losses (I^2 * Z)         0.00              0.00
Transformers        0     Branch Charging (inj)     -                0.0
Inter-ties          0     Total Inter-tie Flow     0.0               0.0
Areas               1

                          Minimum                      Maximum
                  -------------------------  --------------------------------
Voltage Magnitude   1.000 p.u. @ bus 1          1.000 p.u. @ bus 1
Voltage Angle     -13.60 deg   @ bus 4         20.61 deg   @ bus 11
Lambda P            0.00 $/MWh @ bus 1          0.00 $/MWh @ bus 1
Lambda Q            0.00 $/MWh @ bus 1          0.00 $/MWh @ bus 1

================================================================================
|     Bus Data                                                                 |
================================================================================
 Bus      Voltage          Generation             Load          Lambda($/MVA-hr)
  #   Mag(pu) Ang(deg)   P (MW)   Q (MVAr)   P (MW)   Q (MVAr)      P         Q
----- ------- --------  --------  --------  --------  --------  -------  -------
    1  1.000    9.193    456.18      0.00     94.20      0.00     0.000      -
    2  1.000   -8.191       -         -       80.40      0.00     0.000      -
    3  1.000   -1.019      0.00      0.00    145.10      0.00     0.000      -
    4  1.000  -13.600       -         -      206.90      0.00     0.000      -
    5  1.000  -13.372       -         -     1477.50      0.00     0.000      -
    6  1.000    0.000*  1125.00      0.00    225.70      0.00     0.000      -
    7  1.000  -10.478       -         -       66.80      0.00     0.000      -
    8  1.000   -8.765       -         -      308.30      0.00     0.000      -
    9  1.000   14.321    783.25      0.00     28.60      0.00     0.000      -
   10  1.000  -12.914       -         -       71.30      0.00     0.000      -
   11  1.000   20.613    666.67      0.00    174.20      0.00     0.000      -
   12  1.000   10.668       -         -       29.10      0.00     0.000      -
   13  1.000    5.826       -         -       48.80      0.00     0.000      -
   14  1.000   10.105       -         -       31.30      0.00     0.000      -
   15  1.000    4.404       -         -       34.00      0.00     0.000      -
   16  1.000   11.438    460.00      0.00    302.60      0.00     0.000      -
   17  1.000    2.667       -         -       67.20      0.00     0.000      -
   18  1.000    5.514       -         -       13.30      0.00     0.000      -
   19  1.000    9.210       -         -       50.90      0.00     0.000      -
   20  1.000    8.203       -         -       34.90      0.00     0.000      -
                         --------  --------  --------  --------
               Total:    3491.10      0.00   3491.10      0.00
```

```
================================================================================
|     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     10      4     12.16      0.00    -12.16      0.00     0.000      0.00
   2      5     10    -14.25      0.00     14.25      0.00     0.000      0.00
   3      3      6    -79.46      0.00     79.46      0.00     0.000      0.00
   4      6      2    178.11      0.00   -178.11      0.00     0.000      0.00
   5      5      8   -121.80      0.00    121.80      0.00     0.000      0.00
   6      6      5    641.73      0.00   -641.73      0.00     0.000      0.00
   7      7     15   -248.43      0.00    248.43      0.00     0.000      0.00
   8      5     17   -277.03      0.00    277.03      0.00     0.000      0.00
   9     10      2    -97.71      0.00     97.71      0.00     0.000      0.00
  10      8      9   -430.10      0.00    430.10      0.00     0.000      0.00
  11      1      9    -76.51      0.00     76.51      0.00     0.000      0.00
  12      7      4    181.63      0.00   -181.63      0.00     0.000      0.00
  13     20     12    -16.30      0.00     16.30      0.00     0.000      0.00
  14     20     13     28.96      0.00    -28.96      0.00     0.000      0.00
  15      1      5    435.80      0.00   -435.80      0.00     0.000      0.00
  16     11     18    252.49      0.00   -252.49      0.00     0.000      0.00
  17     12     19     30.06      0.00    -30.06      0.00     0.000      0.00
  18      9     15    172.59      0.00   -172.59      0.00     0.000      0.00
  19     14     19     40.68      0.00    -40.68      0.00     0.000      0.00
  20     16     15    109.83      0.00   -109.83      0.00     0.000      0.00
  21      4      5    -13.11      0.00     13.11      0.00     0.000      0.00
  22     19     13     19.84      0.00    -19.84      0.00     0.000      0.00
  23     14     11   -151.26      0.00    151.26      0.00     0.000      0.00
  24     17      3     65.64      0.00    -65.64      0.00     0.000      0.00
  25      1     11    -88.72      0.00     88.72      0.00     0.000      0.00
  26     14     18     79.29      0.00    -79.29      0.00     0.000      0.00
  27     20     16    -47.57      0.00     47.57      0.00     0.000      0.00
  28      9     12     75.46      0.00    -75.46      0.00     0.000      0.00
  29     17     18   -318.47      0.00    318.47      0.00     0.000      0.00
  30      1     17     91.40      0.00    -91.40      0.00     0.000      0.00
                                                              --------  --------
                                                    Total:      0.000      0.00

================================================================================
|     Generation Constraints                                                   |
================================================================================
 Gen   Bus                  Active Power Limits
  #     #     Pmin mu     Pmin       Pg       Pmax     Pmax mu
----  -----   -------   --------  --------  --------  -------
   2     3      0.000    560.00       -     1845.00      -
   3     6        -      260.00   1125.00   1125.00     0.000
   6    16      0.000    460.00    460.00   1215.00      -

Generation cost: $23854
Load shed penalty: $0
Total cost: $23854

Maintenance: Updated Failure Probability
Line F_BUS  T_BUS  Previous  New
28   9      12     0.066     0.013

% Future steps are conducted in a similar manner.
```

Function-Based Example

This appendix next provides an illustrative example of interacting with the simulation via the function-based interface. We will base this example on a function called `benchmark_approach`

that takes four arguments, `mpcs`, `ncrew`, `currentTurn`, and `lastTurn`, and has three outputs, `Pg`, `Pd`, `crewAllocation`. This function is provided below.

```
function [Pg,Pd,crewAllocation] = benchmark_approach(mpcs,ncrew,currentTurn,lastTurn)
% Solve a security-constrained unit commitment at every turn.
% Randomly choose which lines to repair and maintain, weighted by their
% outage statuses and failure probabilities.

%% Set some useful constants
define_constants;
nbus = size(mpcs{1}.bus,1);
ngen = size(mpcs{1}.gen,1);
nbranch = size(mpcs{1}.branch,1);

%% For reference, load in the parameters that control the simulation
[opts, ~, ~] = setParameters;

% Controls what fraction of an outaged line is repaired every turn
repairEffectiveness = opts.repairEffectiveness;

%% Formulate the set of contingencies to consider
% Don't consider any contingencies (N-0 security):
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% contingencies = [];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Consider contingencies consisting of the individual failures for each
% line (N-1 security):
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
contingencies = eye(nbranch,nbranch);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Consider contingencies consisting of the individual failures for each
% pair of lines (N-2 security):
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% branch_pairs = nchoosek(1:nbranch,2);
% contingencies = zeros(nbranch,size(branch_pairs,1));
% contingencies(sub2ind(size(contingencies),branch_pairs(:,1),(1:size(branch_pairs,1)).')) = 1;
% contingencies(sub2ind(size(contingencies),branch_pairs(:,2),(1:size(branch_pairs,1)).')) = 1;
% contingencies = [eye(nbranch,nbranch) contingencies]; % Include all N-1 and N-2 contingencies.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Solve a security-constrained unit commitment and prepare the result for output
[sol] = rundcscuc(mpcs{currentTurn},contingencies);
Pg = sol.gen(:,PG);
Pd = sol.bus(:,PD);

%% Determine the repair crew allocations according to a simple rule
% Simple rule for repair crews: Allocate crews randomly to each outaged
% line, weighted by how close they are to being repaired.
crewAllocation = zeros(ncrew,1);
outaged_lines = find(mpcs{currentTurn}.branch(:,BR_STATUS) ~= 1);

% Randomly order outaged lines, weighted by their outage status such that
% lines which are nearly repaired are assigned crews first.
weighted_status=mpcs{currentTurn}.branch(outaged_lines,BR_STATUS).*rand(length(outaged_lines),1);
[~,order] = sort(weighted_status ,'descend');
ordered_outaged_lines = outaged_lines(order);


% Allocate crews to repair lines. Attempt to have a sufficient number of
% crew allocated to repair a line in one turn.
crewIdx = 1;
for lineIdx = 1:length(ordered_outaged_lines)
   if sum(crewAllocation ~= 0) < ncrew
       % Get the line's status
```

```
        lineStatus = mpcs{currentTurn}.branch(ordered_outaged_lines(lineIdx),BR_STATUS);

        % Compute how many crews are needed to repair this line in one turn.
        ncrewRequired = ceil( (1-lineStatus) / repairEffectiveness);

        % How many of the available crews aren't yet allocated?
        ncrewAvailable = ncrew - sum(crewAllocation ~= 0);

        % Allocate up to this number of crews.
        crewsToAllocate = min(ncrewRequired,ncrewAvailable);
        crewAllocation(crewIdx:crewIdx + crewsToAllocate - 1) = ordered_outaged_lines(lineIdx);

        % Increment the number of crews allocated
        crewIdx = crewIdx + crewsToAllocate;
    else
        break;
    end
end

%% Allocate the remaining crews to do maintenance according to a simple rule
% Simple rule: Have remaining crews perform maintenance, one crew per line,
% in a randomly selected order that is weighted by failure probabilities.

% Get the indices of the lines in order of decreasing failure probability
failure_probability = mpcs{currentTurn}.failure_probability;
[~,failure_sort] = sort(failure_probability,'descend');

for i=1:nbranch
    if sum(crewAllocation ~= 0) < ncrew
        crewAllocation(crewIdx) = failure_sort(i);
        crewIdx = crewIdx + 1;
    else
        break;
    end
end
```

To evaluate this function in the simulation, call the function `power_system_pandemic` with the first argument specifying the name of your function (in this case, the function `benchmark_approach` defined above). This will run the simulation based on the function you provided and display the resulting solution in both the command window and the GUI. To prevent the code from showing the GUI window, set the second argument, `display_gui`, to false.

```
display_gui = false;
[mpcs,Pd,mpcs_specified,Pd_specified,crewAllocation,ncrew,score] ...
     = power_system_pandemic('benchmark_approach',display_gui);
```