Senior Capstone

May 2025

User Guide for Transmission Line Outage Planning Optimizer

Brandon Blankenship

Nate Winebarger

Ariel Colón Rodríguez

Contents

Overview

This library is a tool designed to aid in the scheduling of a system of transmission line outages on a predetermined power system based on changing load and generation data. Features include:

1. N-1 Contingency Analysis.
2. Voltage Magnitude limit checking of buses.
3. MVA Limit checking of branches.
4. Generation outage support.
5. Block generation dispatch support.
6. Linear generation dispatch support.
7. Variable duration up to full year.
8. Parallel computing support by MATLAB parallel toolbox.
9. Robust debugging information for power flow results.

All components of this library are capable of independent operation. All things considered, this is a library of functions that can be tied together to operate as a planning tool.

Installation requirements

This library requires the installation of the following modules:

1. MATLAB
2. MATPOWER
3. Python 3.10
4. Python Libraries
    a. Et_xmlfile
    b. Matlabengineforpython
    c. Numpy
    d. Openpyxl
    e. Pandas
    f. Pillow
    g. Python-dateutil
    h. Pytz
    i. Setuptools
    j. Six
    k. tzdata

Initialization.m

This function serves as the tie in between the Python GUI and the MATLAB backbone of the powerflow calculations. Performed inside this function is a set of setup variables utilized by the rest of the library.

Variable	Usage
sim_settings	Settings object to share settings information from the GUI with other internal functions of the library.
mpopt	MATPOWER powerflow options structure.
mpc	Loaded test case for powerflow operations.
loaddata	Up to year-long load data from “HourlyLoad.xlsx”
Generation_outages	List of generation outages to be included in the simulation. Loaded from “RequiredOutages.xlsx”
Gen_array	List of generation blocks for use with generation block dispatch support.
Ini_results	Results array from initial N-1 contingency to be used by the scheduling algorithm to determine available times for outage scheduling.
Ini_failure	Array showing Boolean success or failure on a per hour basis of the initial N-1 contingency to be used by the scheduling algorithm to determine available times for outage scheduling.
Branch_outages	
schedule	The generated schedule from the scheduling algorithm.

Limits_check.m

Overview

A function to check a snapshot of a power flow simulation for any Voltage Magnitude (VMAG) or Power flow (MVA) limit failures on the system.

Function Call

[limit_check_return, failure_params] = limits_check(mpc_case)

Inputs

Mpc_case	The powerflow result from N1_contingency.m. See N1_contingency.m for more information.

Outputs

Limit_check_return	Boolean value indicating the success or failure of the limit check. Both VMAG and MVA limits must succeed in providing a success flag on this variable to yield a passing state. Values: 1. True = No failures detected for VMAG or MVA limits 2. False = Failure detected for either VMAG or MVA limits, or both.
Failure_params	A structure containing information on the following: 1. Bus location(s) of VMAG failures. 2. Branch location(s) of MVA failures. 3. Magnitude value(s) of VMAG failures. 4. Magnitude value(s) of MVA failures.

Load_data.m

Overview

A class object used to store load data from a formatted excel sheet. This facilitates easier data sharing between functions in the confines of the MATLAB power flow simulation. This can be used as a standalone object, but does not possess many special functions and should only be used as it is already used in the designed solution.

This functionality was intended to be used in future features with being able to add load. It is not fully integrated into the power flow simulation, and as such the internal functions require more work to be done to have an effect on the power flow results.

Constructor Call

Load_data = load_data(settings)

Internal Functions

Add_load(hour, extra_load)

- Adds additional load to the load_data object for all hours looking forward from the passed variable “hour”

Remove_load(hour, lesser_load)

- Removed load from the load_data object for all hours looking forward from the passed variable “hour”

Internal Properties

weighted_load(hour)

- Returns the weighted load for an hour index relative to the peak load. For example, the peak hour index would return 1.

actual_load(hour)

- Returns the actual load in megawatts for an hour index.

Local_settings.m

Overview

A MATLAB class object used to store settings data from settings.txt (the settings file created by the groups GUI solution). This is hard coded to be created off a text file named 'settings.txt'. As such there is no constructor or function calls that should be used as such.

Internal Parameters

verbose

- Integer value of 0, 1 or 2. Used to control pretty print output of MATPOWER to lower the spamming of information during the power flow simulation.

outage_sheet

- String name of outage excel sheet being used in the power flow simulation. "RequiredOutages.xlsx" in this case study.

load_sheet

- String name of the load data sheet being used in the power flow simulation. "HourlyLoad.xlsx" in this case study.

algorithm

- String name of the power flow algorithm used in the power flow algorithm. "NR-SH" in this case study. Can be used with any algorithm used by MATPOWER.

case_name

- String name of the case being used in the power flow simulation. "case118_CAPER_PeakLoad.m" in this case study.

end_hour

- Integer value of the ending hour of the simulation. 8760 for most use cases.

start_hour

- Integer value of the starting hour of the simulation. 1 for most use cases.

case_sheet

- String name for the excel sheet containing the original case data. "InitialCaseData.xlsx" for this case study.

block_dispatch

- Integer value of 1 or 0 used to set block dispatch functionality. 1 is enabled and 0 is disabled.

generation_outage

- Integer value of 1 or 0 used to set generation outage functionality. 1 is enabled and 0 is disabled.

N1_contingency.m

Overview

A function used to run an N-1 contingency analysis over a range of hours. This function runs on the MATPOWER library and requires MATPOWER to be installed to be used. Information about the state of the system in the confines of the N-1 contingency is returned in the return variables. Partial runs of the year will return arrays starting from 1 regardless of the value used for start_hour.

Function Call

[results_array, failure_array] = n1_contingency(settings, scheduled_outage, generation_outages, load_data, mpc, gen_array, mpopt, start_hour, end_hour)

Inputs

Settings	Used to pass in the settings object to pass in the state of block dispatch and generation outages.
Scheduled_outage	A class object to run N-1 analysis for transmission outages. See: “scheduled_outage.m”
Generation_outages	A class object containing data for the generation outages present in the simulation. See: “generation_outage.m”
Load_data	A class object containing data for varying load data for the simulation. See: “load_data.m”
Mpc	Baseline MATPOWER results passed in. Should be the baseline at peak load.
Gen_array	An array used to pass in generation block information for block dispatch functionality.
Mpopt	Power flow options for MATPOWER power flow simulation.
Start_hour	The start hour of the N-1 window.
End_hour	The end hour of the N-1 window.

Outputs

Results_array	Array of the length of the number of hours the N-1 analysis is ran. Contains 0’s and 1’s. 0’s signify some failure at that hour index, 1’s signify a success at the hour index.
Failure_array	Array containing failure information for any hour returning a 0 for results_array. Contains

	the information returned by the limits_check.m function.

Priority.m

Overview

The function evaluates and ranks branch outages based on various operational and scheduling factors. It reads outage data from an Excel sheet, converts all time-related values to a common unit (hours), and calculates a priority score for each outage using criteria like duration, power flow impact, scheduling window, and system dependencies.

The output is a structured list of outages sorted from highest to lowest priority. This helps planners quickly identify which outages are most critical or time-sensitive, enabling more efficient and informed scheduling decisions

Function Call

[branch_outages] = priority(sim_settings)

Inputs

sim_settings	Settings object to share settings information from the GUI with other internal library functions.
RequiredOutages.xlsx	It reads the Transmission Outages sheet for all the outages that are wanted and analysis them one by one

Outputs

branch_outages	The output provides a detailed prioritization of outages, listing key information such as branch number, duration, timing windows, dependencies, overlap requirements, and system status. While not a finalized schedule, this output serves as the foundation for informed scheduling decisions by identifying constraints, coordination needs, and readiness for each outage.

Runpf_wcu.m

Overview

This function is mostly kept the same as the baseline runpf.m from MATPOWER. The change made is at line 506 with the below section of code:

```
506   %%This line removes table vomit from the output window
507   if (mpopt.verbose > 0)
508       printpf(results, 1, mpopt);
509   end
510
```

This removes all of the power flow solving information that is printed to the MATLAB window by default.

Outside of the above change, this is just the runpf.m file from MATPOWER and the MATPOWER documentation should be referenced for its use.

Schedule_algorithm.m

Overview

This function schedules transmission outages based on a priority list and an initial N-1 contingency case. It starts by attempting to find a time slot for each outage, considering its priority, duration, and any dependencies or overlaps. Once a potential slot is found, the function runs another N-1 contingency check to ensure that the outage schedule is successful and doesn't introduce any failures in the system. If the check is successful, the outage is finalized in the schedule, and the function moves on to the next outage. If the check fails, the function attempts to find an alternative slot for the outage. If no valid slots are found, the case is marked as failed, and the function proceeds to the next outage in the priority list. This iterative process ensures that only feasible outage schedules are included in the final plan.

Function Call

[schedule] = schedule_algorithm(settings, branch_outages, initial_results_array, generation_outages, load_data, mpc, gen_array, mpopt)

Inputs

settings	Settings object to share settings information from the GUI with other internal library functions.
branch_outages	A detailed prioritization of outages, listing key information such as branch number, duration, timing windows, dependencies, overlap requirements, and system status.
initial_results_array	Results array from the initial N-1 contingency to be used by the scheduling algorithm to determine available times for outage scheduling.
generation_outages	A class object containing data for the generation outages present in the simulation
load_data	A class object containing data for varying load data for the simulation.
mpc	Loaded test case for powerflow operations.
gen_array	Array of information about the generation block allocation
mpopt	MATPOWER powerflow options structure.

Outputs

schedule	A class storing outage data for each branch (start/end hours, success/failure)

Scheduled_outage.m

Overview

A MATLAB class object used to store outage data for running the scheduled transmission outages from the scheduling algorithm function.

Constructor

Outage = scheduled_outage(start_hour, end_hour, branches)

Success will always default to true and does not need to be set with the constructor.

Internal Parameters

start_hour

- Integer value that signifies the start hour of the year for the outage.
- Usage
  - Hour = outage.start_hour
  - Sets Hour equal to the start hour of the outage.

end_hour

- Integer value that signified the end hour of the year for the outage.
- Usage
  - Hour = outage.end_hour
  - Sets Hour equal to the end hour of the outage

Branches

- Array of branches that are taken offline during an outage. This can be one or multiple values.
- Usage
  - Branches = outage.branches
  - Sets branches equal to the array of values stored in the outage object.

Success

- Boolean value to signify either the success or failure of an outage after N-1 contingency analysis.
- Usage
  - Success = outage.success

- Sets success equal to the success state of the outage object.

Internal Functions

Success = set_state(this, state)

Usage:

Calling a scheduled outage with true or false as an input to change the success flag of the outage.

Example:

outage.set_state(false)

set the success flag of outage to false.

Case118_CAPER_PeakLoad.m

This file is identical to the file provided by CAPER at the start of the project except for one change at line 222. The MVA rating of the branch was uprated from 500 MVA to 600 MVA. All other functionality of this file remains the same as originally handed over to the team.

Gen_scale_block.m

Overview

This function provides generation block dispatch functionality to the design. The blocks are based on the block information provided by CAPER and are set based on the dispatch block from the InitialCaseData.xlsx file provided by CAPER.

Note that islands should be removed from the system before the use of this function when used in an iterative approach.

This function operates by calculating the total load present on the gen_mpc variable that is passed in, then allocating on a block by block basis the generation required to meet this load demand.

Function Call

Scaled_generation = gen_scale_block(gen_mpc, gen_array)

Inputs

Gen_mpc	The power flow solution that requires scaling. Must contain a gen struct that will be scaled.
Gen_array	Array of information about the generation block allocations that is created in initialization.m by the generation_block.m file.

Outputs

Scaled_generation	Scaled form of gen_mpc that is returned to represent a scaled form of the system.

Generation_block.m

Overview

A MATLAB class object used to store information related to generation blocks as dictated by the input documents.

Constructor

block = generation_block(settings, block)

Settings is the settings class from local_settings.m.

Block is the numerical block between 1 and 5 (inclusive).

Internal Parameters

busses

- Array of the generation bus locations present in the block
- Usage
  - bus = generation_block.busses(index)
  - returns the generation bus location at the index location.

capacity

- Integer value that signifies the capacity at individual generation bus locations.
- Usage
  - capacity = generation_block.capacity(index)
  - returns the capacity at the generation bus at the index location.

Total_capacity

- Integer value of the total capacity of the block.
- Usage
  - Total_capacity = generation_block.total_capacity
  - Returns the total capacity present on the bus.

Generation_outage.m

Overview

A MATLAB class object used to store information related to generation outages for the simulation.

Constructor

outage = generation_outage(start_hour, duration, bus, real_power)

Note that duration should be the number of hours that the outage lasts for as the constructor will add it to the start_hour parameter to derive the end_hour parameter.

Internal Parameters

Start_hour

- Integer value signifying the start hour of the year for the generation outage.
- Usage
  - start = generation_outage.start_hour
  - returns the hour of the year that the generation outage starts on.

End_hour

- Integer value that signifies the ending hour of the year for the generation outage.
- Usage
  - end = generation_outage.end_hour
  - returns the hour of the year that the generation outage ends on.

bus

- Integer value of the generation bus location that the outage is occurring at.
- Usage
  - Bus = generation_outage.bus
  - Returns the bus location for the generation bus that the outage is located at.

Real_power

- Integer value for the real power generation (P value) for the generation bus that is to be taken out of service.
- Usage
  - Power = generation_outage.real_power
  - Returns the real power (P value) for the generation outage.

HourlyLoad.xlsx

This sheet has minor modifications from the original sheet provided by CAPER. Column E has been added to calculate a weighted value of the max load relative to the peak load present on the system. H2, H3 and I3 are also added to showcase the max load, average load and average weighted load present in the system. None of these cells are referenced to and can be left out of future use cases.

This sheet is used heavily by the simulation for scaling load and generation. The format must remain the same as it is in the file package turned over at the end of this project.

InitialCaseData.xlsx

This sheet has no modifications from the original sheet provided by CAPER. It is used by the simulation and is required for any other cases. The format must remain the same as it is in the file package turned over at the end of this project.

RequiredOutages.xlsx

The generation tab has had the block dispatch information removed as the project instead pulls this information from the InitialCaseData.xlsx file. Otherwise, the generation tab is the same and must remain in the format as it is in the file package turned over at the end of this project.

The format of the Transmission tab effectively listed the physical connections ('From Bus' to 'To Bus') involved in each task, along with the estimated 'Duration' and a catch-all 'Other Considerations' field. While this provided a good overview of the work and any immediate notes, it presented challenges for systematic planning and coordination.

To address these limitations, the data was structured differently. By breaking down the 'Other Considerations' into dedicated columns like 'Starting Time Frame,' 'Ending Time Frame,' 'Dependency,' 'Outage Overlap,' and 'Independency,' we gained a much clearer and more actionable view. This structured format allows for better scheduling, resource allocation, identification of critical paths, and understanding of potential conflicts or dependencies between different work items. The addition of fields like 'Bus Status,' 'Load (MW),' and 'Line Uprate (MVA)' also provides a dedicated space to track the operational impacts of the planned work, which was less clearly defined in the original format.

ScheduleOrder.xlsx

This sheet is an output of the program which details the transmission outages that are automatically scheduled by the tool.

Settings.txt

This sheet is an output of the program which provides setting information to the simulation program. The sheet provided in the file package at the end of the project are the default values used by the team during the testing and design of the project. The formatting of this sheet is very important and all information must be located on the lines that are present when this file package is turned over at the end of the project.

MyApp.py

Overview

This application is a graphical user interface (GUI) built using the Tkinter Library in Python and is integrated with MATLAB to assist in running power system outage planning simulations. It allows users to load required Excel files, configure simulation settings, and execute MATLAB code.

How to use application

In line 66 of the Python code make sure to change where your file directory is for your MATLAB code and other files.

self.matlab_script_directory = r"C: File path here"

1. When Running Python with the MyApp.py code, run the code
2. The GUI will pop up and you have two tabs in the top left Main and Settings
    2.1. Main tab is where you will load the respective excel sheets of data. Click and load through the file explorer to load the sheets.
    2.2. In the settings tab choose what settings you wish to run
3. When finished inputing excel sheets and settings click the Start button on the Main tab to run the MATLAB code. (This code will take a long time to run)

Function Call

Inputs

open_file_dialog	Allows the user to select Excel files via a file dialog box. It also performs header and column validation against predefined formats (EXPECTED_HEADERS, EXPECTED_COLUMNS).
create_excel_drop_area	Sets up clickable GUI drop zones for each Excel file type. These zones are linked to open_file_dialog.
toggle_block_dispatch	Toggles the Block Dispatch setting (On/Off) and updates the button label accordingly.
toggle_generator_outages	Toggles the Generator Outages setting (On/Off) and updates the button label.
create_settings_file	Collects user-selected settings from dropdowns and entries (e.g. verbose level,

	hours, method), formats them, and writes a settings.txt file for MATLAB.
time_entry.get() / start_hour_entry.get()	Retrieves the numeric input for simulation time and start hour from the GUI entry widgets.
verbose_combobox.get() / method_combobox.get()	Retrieves the user's selection of verbosity level and algorithm type.

Outputs

append_text	Appends output text (logs or status updates from MATLAB) into a read-only text widget in the GUI. This is called by the TCP listener thread.
handle_incoming_data	Called when new data is received from the TCP connection; it ensures thread-safe GUI updates by scheduling append_text on the main thread.
run_matlab_code	Runs the MATLAB script initialization.m in a separate thread, but only if settings.txt exists. Clears the output widget first.
_run_matlab_script	Executes the MATLAB script initialization.m using the MATLAB engine. Catches exceptions and logs them to console.
load_excel_data	Copies validated Excel files into the MATLAB working directory. Provides success/failure message popups.
create_settings_file	Generates the formatted settings.txt used by the MATLAB script based on the user's inputs. Writes to disk and shows a success or error popup.

outageCalendarGUI.m

Overview

The script creates an interactive MATLAB GUI that visualizes outage schedules across an annual calendar interface. It reads outage data from an Excel file and displays color-coded monthly and daily views, where colors represent the number of outages per day. Users can click on a month to view its daily breakdown and select individual days to inspect specific outage details, including hourly timing and identifiers.

This visual tool supports outage planning by helping users quickly identify peak outage periods, compare monthly outage patterns, and explore outage timing at a daily level. Its intuitive layout and interactive navigation improve situational awareness and aid in effective scheduling decisions.

Function Call

outageCalendarGUI(initial_results_array)

Inputs

Initial_results_array	Results array from the initial N-1 contingency to be used by the scheduling algorithm to determine available times for outage scheduling.
ScheduleOrder.xlsx	It reads the Transmission Outages sheet for all the outages that are wanted and analyzes them one by one.

Outputs

The output of the GUI is a dynamic, color-coded calendar interface that visually displays the outage schedule across a full year. Each month is represented by a button, and selecting a month opens a detailed daily view, where each day is shaded based on the number and severity of scheduled outages, ranging from normal operation to critical system failures. Clicking on a specific day reveals an hourly breakdown of branch outages, offering precise insight into when and where each outage occurs. This output provides an intuitive and interactive way to assess outage distribution over time, making it easier to identify trends, peak outage periods, and potential scheduling conflicts.

Future Work

1. Generation dispatch can use some work as it has issues with generation outages in this case study. It is possible that dispatching generation based on location will fix these issues in this case study.
2. The scheduling algorithm output can be moved to python in order to display the results in a better fashion that MATLAB allows.
3. The power flow part of the design can be moved to python, C/C++, C# or FORTRAN to get off the MATLAB license requirement being a cost to take into account for the tool.
4. An extension to part 3 is writing a power flow system more tailored to this particular use case. Embedding generation and load scaling into the design may allow for more computation optimization.
5. Functionality can be added to add or remove outages during a simulation.
6. Possible optimization is to split the year up into smaller chunks as opposed to running the full year in one go at the start. This would allow for the scheduling of some outages that are required to be early in the year. For example, run Jan 1 through April 1 to analyze and schedule outages that must be completed by April 1.
7. Adding a feature to generate the available times for scheduling then letting the user manually drop outages into the spots and then simulating off that schedule is a possible feature that could be added.