PyEMTG User Guide

Noble Hatten[*], Edwin Dove[*]

Revision Date	**Author**	**Description of Change**
July 22, 2022	Noble Hatten	Initial creation.
Sept. 19-20, 2022	Noble Hatten	added PEATSA section.
March 26, 2024	Edwin Dove	Added PyEMTG GUI navigation text.

[*]Aerospace Engineer, NASA Goddard Space Flight Center, Navigation and Mission Design Branch Code 595

Contents

List of Acronyms

EMTG Evolutionary Mission Trajectory Generator

GUI Graphical User Interface

PEATSA Python EMTG Automated Trade Study Application

1 Introduction

The purpose of this document is to describe:

- How to prepare to use the PyEMTG Graphical User Interface (GUI) for Evolutionary Mission Trajectory Generator (EMTG): Section 2.1.
- How to Navigate in the EMTG PyEMTG GUI: Section 2.
- How to use the Python EMTG Automated Trade Study Application (PEATSA) utility: Section 3.

2 PyEMTG GUI

PyEMTG GUI is a graphical user interface (GUI) written in Python which is used to process EMTG input and output files. It provides users easier access to most of the Core EMTG application functionality and contains EMTG post-processing functionality.

2.1 Preparing to Use the PyETMG GUI

In the PyEMTG directory of the EMTG repo, there is a file called PyEMTG-template.options. This file must be duplicated, and the copy must be renamed PyEMTG.options. The new PyEMTG.options file contents must be edited to reflect the user's system.

The PyEMTG.options file consists of multiple lines of text. Each line consists of space-delimited name value pairs, where the left block of text is the name/variable and the right block of text is the value associated with that name/variable. The most important variable to set is `EMTG_path`. `EMTG_path` is important because setting it appropriately allows for the EMTG executable to be executed by the PyEMTG GUI. `EMTG_path` must be set to the full path and name of the EMTG executable file. All paths in the options file must use forward slashes as the file/folder separators. For example, if the EMTG repo were located in C:\emtg, then the user would type:

```
EMTG_path C:/emtg/bin/EMTGv9.exe
```

The other variables in the PyEMTG.options file are optional because the default values set in the PyEMTG.options file can be (and almost always are) overridden in the options file for a given EMTG mission. However, it should be noted that the default EMTG mission will not run unless either the default values in PyEMTG.options are set appropriately or the relevant values are set appropriately in the options file for the default mission.

Once the PyEMTG.options file has the required EMTG executable configured, in the prompt the user would need to navigate to the PyEMTG folder, type the following, and then press enter to Launch PyEMTG:

```
python PyEMTG.py
```

2.2 Navigating the PyETMG GUI

Once in the PyEMTG GUI a user is presented with a screen where they can use the File menu to either open an existing mission via an EMTG options file or create a new mission. All PyEMTG GUI functionality can be accessed through a user's keyboard and mouse or soley using the keyboard. The ability to solely use a keyboard to interact with the PyEMTG GUI should also allow it to be compatible with assitive technology devices. The selections of the File menu contains keyboard shortcuts that a user can use for quick access to the File menu selections.

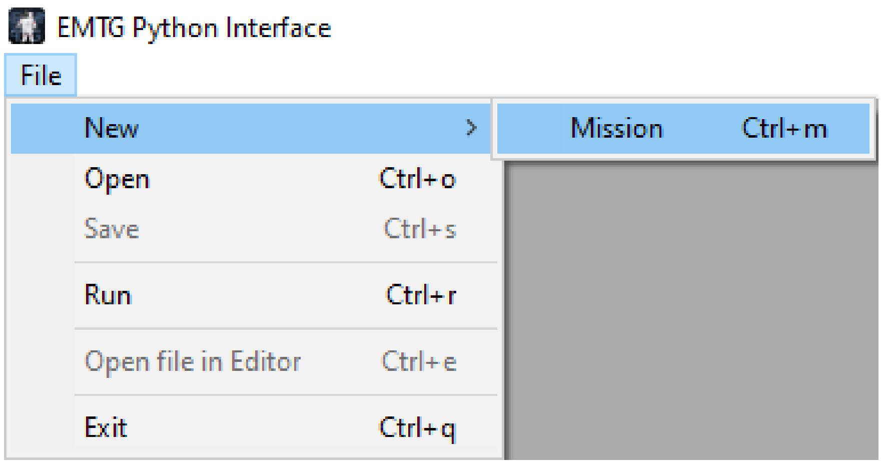
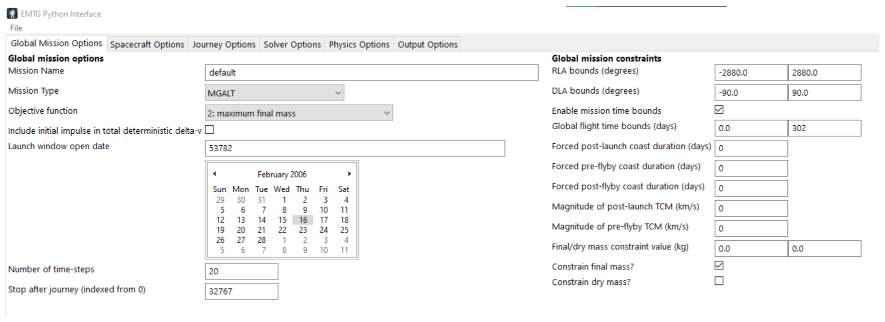


Figure 1: EMTG File Menu

Once an EMTG mission is populated in the GUI, several tabs become visible to the user where they can alter values through several input fields (i.e. calanderCTRL pickers, text fields, check boxes, drop-down menus).



Global mission options

Mission Name	default
Mission Type	MGALT
Objective function	2: maximum final mass
Include initial impulse in total deterministic delta-v	☐
Launch window open date	53782
Number of time-steps	20
Stop after journey (indexed from 0)	32767

February 2006

Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	1	2	3	4
5	6	7	8	9	10	11

Global mission constraints

RLA bounds (degrees)	-2880.0	2880.0
DLA bounds (degrees)	-90.0	90.0
Enable mission time bounds	☑	
Global flight time bounds (days)	0.0	302
Forced post-launch coast duration (days)	0	
Forced pre-flyby coast duration (days)	0	
Forced post-flyby coast duration (days)	0	
Magnitude of post-launch TCM (km/s)	0	
Magnitude of pre-flyby TCM (km/s)	0	
Final/dry mass constraint value (kg)	0.0	0.0
Constrain final mass?	☑	
Constrain dry mass?	☐	

Figure 2: EMTG Global Mission Options Tab

The names of the tabs are as followed:

- Global Mission Options
- Spacecraft Options
- Journey Options
- Solver Options
- Physics Options
- Output Options

The following are keyboard sequences that allow a user to navigate through the GUI tabs:

- Ctrl + Tab: Go to the next GUI Tab
- Ctrl + Shift + Tab: Go to the previous GUI Tab
- Shift + Tab: Go to a previous input field
- Tab: Go to the next input field

Once a user enters the calendarCTRL input field, the following keyboard sequences allow a user to select a date and navigate:

- arrow keys: Move to a new day/Month/Year/Year-Range
- Tab: Exit calendar input field to next input field
- Shift+Tab: Exit calendar input field to previous input field
- Ctrl + up arrow: Zoom out upper calendar selection to be broader (e.g. going from month to year, year to year-range)
- Ctrl + down arrow: Zoom in upper calendar selection to be narrower
- Ctrl + left/right arrow: Move to next value of the upper calendar selection

3 PEATSA

PEATSA is a set of Python scripts used to create and execute EMTG trade studies. PEATSA operates by taking an EMTG options file (the "base case"[1]) and executing variations of some of those options. The user sets the different variations prior to exectution. For example, a user might vary the launch date, the launch vehicle, a gravity-assist sequence, and/or a maximum time of flight. One execution of all desired variations constitutes a "PEATSA iteration." At the end of each iteration, a user-defined objective function is evaluated for each EMTG case, and this value is compared against the best objective function value achieved in any previous iteration for each EMTG case. The EMTG execution with the best value of the objective function for each case is saved, and information about the best execution for each case is written to a summary file.

For the next iteration, initial guesses for EMTG cases are generated based on existing results from cases with similar option variations. For example, if the user is varying the launch date (i.e., the launch date is a trade parameter), then the user can allow a case to use an initial guess solution from another case (a "neighbor") as long as the launch dates of the two cases are within the vicinity of ∼5 days of each other, and all other trade parameters are the same between the two cases. This process is called "seeding" or using one case to "seed" another. The idea behind seeding is that similar solutions are likely to exist between neighboring cases, so using a neighbor as a seed is likely to result in a feasible solution. In this way, a solution for a single case can turn into a "family" of solutions after a number of iterations as neighbor i seeds neighbors $i+1$ and $i-1$ on iteration j, and then neighbor $i+1$ seeds neighbors i and $i+2$ on iteration $j+1$. Note that in the preceding description, a neighbor that is used as a seed may also be seeded by a neighbor of its own. As a result, any new optimal solution can "propagate" to create a new solution family.

The remainder of this section decribes how to create, execute, and analyze the results of a PEATSA run.

3.1 Creating a PEATSA Run

At the most fundamental level, three things are required to execute a PEATSA run:

- EMTG options file: This is called the base case and must be able to produce a valid/feasible solution.
- PEATSA options file: This is a Python file that specifies the options to be used when executing the run. This is also known as the "trade study options file".
- PEATSA trade parameters file: This csv file sets the trade parameters of the PEATSA run.

These three items are described in more detail in Sections 3.1.1, 3.1.2, 3.1.3, and 3.1.4.

[1]nomenclature that may be unfamiliar to a non-PEATSA user is written in quotes on first reference.

3.1.1 The Base Case

The base case is an EMTG options file that can produce a feasible solution. The purpose of the base case is to set the default values of the options for an EMTG run that PEATSA will then systematically change in order to perform the trade study. Once a problem requiring a PEATSA run is identified, the most common workflow is to use the PyEMTG GUI to create the base case. In this pre-PEATSA phase, it is beneficial but not absolutely necessary to find a feasible solution for the base case and set the initial guess for the base case to be that feasible solution.

3.1.2 Creating the PEATSA Options File

The PEATSA options file is a Python file that specifies the options to be used when executing the PEATSA run. The PEATSA options file uses Python syntax, but it is not actually an end-to-end script or function. Instead, it is a sequence of variable assignments that is evaluated by PEATSA.

A new PEATSA options file is created by either duplicating and then modifying an existing PEATSA options file or by using the PEATSA_script_generator.py utility script in **<EMTG_root_dir>**/PyEMTG/PEATSA, where **<EMTG_root_dir>** is the top level directory of EMTG that has the /bin/ directory as a sub-directory. To create a new PEATSA options file, execute the PEATSA_script_generator.py script in Python. The script prompts the user to answer a series of questions in order to create the desired PEATSA options file for the run. (Some elements here are not described in detail yet; information will be added later, but the current contents are enough to create a new PEATSA run.)

1. `# How should PEATSA start? ... Enter start_type =`
   - `Fresh`. Used when starting a new PEATSA run. This is the most commonly used option.
   - `Warm`
   - `Hot`
2. `# What type of PEATSA run is this? ... Enter PEATSA_type =`
   - `0`.
   - `1`.
   - `2`. A trade study. This is the option used to systematically vary EMTG options to perform a trade study.
   - `3`.
   - `4`.
   - `5`.
3. `# What is the objective type? ... Enter objective_type =`
   - `0`. A new feasible EMTG solution is considered superior to an old solution if the new solution's PEATSA objective function is better than the old solution's PEATSA objective function.

- `1.`

4. `# Should the default plots be generated? ... Enter generate_default_plots =`

   - `0.` Do not generate default plots. The default plots do not currently work, so this option should be chosen.
   - `1.`

5. `Enter filename:`

   - Full path to and name of file in which to write the new PEATSA script. Filename should end with .py.

After the execution of PEATSA_script_generator.py, the filename entered by the user is populated with options that the user sets to customize the PEATSA run.

3.1.3 Customizing the PEATSA Options File

A PEATSA options file created by executing PEATSA_script_generator.py is not ready to be run. Some options have valid default values that the user may wish to change, while other options *must* be set by the user prior to execution. The listing of options below are a subset of the options available that are important to understand. Refer to the comments in the option file for information on all options available.

- run_name
  Directory for PEATSA to create, which it populates with the files it generates. This option sets the name of that directory. Do not use a file path here.
- working_directory
  Full directory path to the location in which the run_name directory will be created. The path does not need to end with a file separator.
- nCores
  A two element array associated with how PEATSA will split up its processes. Leave the first element of the tuple set to 'local'. Set the second element of the tuple to the number of parallel processes you want PEATSA to use.
- emtg_root_directory
  Full path to the EMTG bin directory.
- if_run_cases
  Set this to 1 to execute the EMTG cases.
- trade_study_options_files
  The list is populated with 2-element tuples. The first element of each tuple is a string containing the full path to a trade parameter csv, and the second element is an integer specifying the type of the trade parameter csv. (See Section 3.1.4.)

- killtime
  This sets a maximum time in seconds that an EMTG case is allowed to run before PEATSA kills the process. The comments in the options file give best practices for setting this value.

- objective_formula
  This sets the PEATSA objective value; that is, how PEATSA decides if one EMTG run is better than another. The formula must be a function of data that is obtainable from a PyEMTG Mission object and MissionOptions object for a given EMTG run.[2] These objects are accessed via 'M.' and 'MO.', respectively (e.g., 'M.total_deterministic_deltav' gives the total deterministic Δv of a solution).
  Note that this objective value does not have to be the same as the objective function for a single EMTG run.

- max_or_min
  Set whether the objective_formula should be maximized or minimized.

- peatsa_goal_objective
  Set appropriately based on the objective_formula. (See the comments in the options file give best practices for setting this value.)

- fingerprint
  The list of strings in the fingerprint defines a unique EMTG case. A good starting point is to make the fingerprint a list of the trade parameter column headers (See Section 3.1.4).
  Note: Disregard the "Do not put any formulas that are already in the seed_criteria list" sentence in the option file comments for this option. It is WRONG.

- seed_folders
  Seeds refers to EMTG solution files (*.emtg files) that can be used as initial guesses for EMTG cases generated by PEATSA. A seed folder contains one or more of these solution files. The seed_folders option is a list of tuples. The first element of the tuple is an integer describing the type of the folder. The second element is a string containing the full path to the seed folder. (See the comments in the options file for additional details.)

- seed_from_seed_folders_on_fresh_start
  Set whether an external seed folder should be used before the first PEATSA iteration. The default value is False, but it is usually more likely that a user wants to set it to True if using seed folders.

- seed_from_cases_that_havent_met_target
  Set whether cases that don't meet the PEATSA goal criteria can be used as a seed in another case. The default value if 0, but it is likely that a user wants to set it to 1.

- seed_criteria
  This option describes how an existing EMTG solution may be used as an initial guess for a new EMTG case with a similar, but not necessarily identical, set of trade parameters. seed_criteria is a list of 4-element tuples. The first element is a string that is the criterion to be compared between an EMTG case and a potential seed case. This should be a trade parameter as defined in the trade parameters csv file. (See Section 3.1.4.) The second element

[2]The Mission object gives information that can only be evaluated *after* an EMTG case is executed, while a MissionOptions object gives information that is known *prior* to EMTG case execution.

is a real number that is the maximum negative seed range. In other words, when evaluating a potential seed case, how much less than the current case can the seed criterion be for the seed case and still be considered a potential seed? The third element is a real number that is the maximum positive seed range. In other words, when evaluating a potential seed case, how much greater than the current case can the seed criterion be for the seed case and still be considered a potential seed? The fourth element is an integer and is the seed selection criterion. The choices are described fully in the comments of the PEATSA options file. Additional notes about seed_criteria:

  - In order for case A to be a potential seed for case B, case A's fingerprint must be identical to case B's fingerprint except for the value of the seed_criteria element under evaluation.
  - The elements of the seed_criteria list are evaluated one at a time. In other words, there is no "and" or "or" logic. Each list element creates seeded cases independently of the others.

- override_options
  A list of two-element tuples used to set EMTG options that are different from the base case but are not trade parameters. The first element of a tuple is a logical condition that is evaluated to see if that specific override should be applied. The second element of a tuple is the formula for that specific override option, of the form 'option = value'. A PyEMTG MissionOptions object is accessed via 'MO'. If a string is required as part of the tuple, then use single quotes to enclose the entire tuple and double quotes to enclose the string element.

  See the following examples of the most commonly used override options:

  - ('1', 'MO.HardwarePath = "/path/to/HardwareModels/"')
  - ('1', 'MO.universe_folder = "/path/to/universe/"')
  - ('1', 'MO.MBH_max_run_time = 120') # or some other value
  - ('1', 'MO.snopt_max_run_time = 10') # or some other value
  - ('1', 'MO.quiet_basinhopping = 1') # or some other value
  - ('1', 'MO.short_output_file_names = 1') # or some other value

- extra_csv_column_definitions
  (See Section 3.3) After each PEATSA iteration, a summary csv file is created. There is a row for each unique EMTG case, as defined by the PEATSA fingerprint. There are columns with data about each case. The extra_csv_column_definitions is used to define columns of the summary csv with data that the user is interested in for that particular PEATSA run. A column is added by adding a two-element tuple to the extra_csv_column_definitions. The first element is a string defining the column header. The second element is a string that may be evaluated for a given EMTG run. Data must be obtainable from a PyEMTG Mission object and MissionOptions object for a given EMTG run. These objects are accessed via 'M.' and 'MO.', respectively. Both Mission and MissionOptions have a 'Journeys' list: a list of Journey objects for Mission and a list of JourneyOptions objects for MissionOptions. In addition, each Journey object has a list of missionevents objects.

  See the following for examples of useful csv columns to include:

  - ('LaunchVehicle', 'MO.LaunchVehicleKey')
  - ('C3 (km2/s2)', 'M.Journeys[0].missionevents[0].C3')

- (‘TotalDeltaV (km/s)’, ‘M.total_deterministic_deltav’)
- (‘DryMass (kg)’, ‘M.spacecraft_dry_mass’)
- (‘TotalFlightTime (years)’, ‘(M.Journeys[-1].missionevents[-1].JulianDate - M.Journeys[0].missionevents[0].JulianDate)/365.25’)
- (‘Flyby 0 GregorianDate (TDB)’, ‘M.Journeys[0].getMGAPhaseUnpoweredFlyby(0).GregorianDate’)
- (‘Phase 0 Arrival Sun Periapse Distance (au)’, ‘M.Journeys[0].GetBoundaryConstraintOutputValueByName (“j0p0MGAnDSMsEphemerisPeggedFlybyIn periapse distance”, “float”) / 1.49597870691e+8’)

3.1.4 The PEATSA Trade Parameters File

The PEATSA trade parameters file contains the path to the base case and sets the trade parameters and their values for the PEATSA run. The trade parameters file is a csv file, and it is recommended to view and edit the file using Microsoft Excel or another spreadsheet tool rather than a text editor.

The first row of the csv contains the full path to the directory containing the base case. The path does not need to end with a file seperator. The path does not contain the name of the actual EMTG options file itself — just the path to the directory in which it resides.

The second row of the csv contains the name of the EMTG options file for the base case. This file must be inside the directory specified on the first row.

The third row of the csv lists, as comma-seperated values, the names of the EMTG options that are trade parameters. The trade parameters are accessed by accessing attributes of the PyEMTG MissionOptions class. Here, a MissionOptions instance for the base case is accessed via ‘MO’. For example, if it is desired to set the maximum total time of flight as a trade parameter, an element of row 3 would be ‘MO.total_flight_time_bounds[1]’. In this example, the index 1 is used because the total_flight_time_bounds attribute is a two-element list containing the lower bound and upper bound of the total time of flight. Another list attribute that is frequently used is ‘Journeys’, which is a list of JourneyOptions objects, one for each Journey of the Mission. For example, ‘MO.Journeys[1].duty_cycle’ would make the duty cycle of Journey index 1 (the indexing starts from 0) a trade parameter.

Rows four and below list the values of the trade parameters. The desired values for a given trade parameter are listed in the column in which the name of that trade parameter was written in row 3. Three different formats are available.

The following are examples of different valid types of trade parameter files in **<EMTG_root_dir>**/PyEMTG/PEATSA:

- Type 0.
  - Trade all values for a given trade parameter individually against the base-case values of all other trade parameters.

  - The number of rows for each column may be different.
  - The total number of cases is equal to the sum of the number of rows of each column, starting with row 4.
  - There is an example in **<EMTG_root_dir>**/PyEMTG/PEATSA/opt_file_type_zero.csv.

- Type 1.
  - Trade all values of each variable against all values of all other variables. An EMTG case is created for every permuation of the values listed in rows 4 and below.
  - The number of rows for each column may be different.
  - The total number of cases is equal to the product of the number of rows of each column, starting with row 4.
  - There is an example in **<EMTG_root_dir>**/PyEMTG/PEATSA/opt_file_type_one.csv.

- Type 2.
  - The columns of each row represent a set of trade parameters that defines an EMTG case to be run.
  - The number of rows for each column must be the same.
  - The total number of cases is equal to the number of rows, starting with row 4.
  - There is an example in **<EMTG_root_dir>**/PyEMTG/PEATSA/opt_file_type_two.csv.

3.2 Executing PEATSA

Once the files required to set up a PEATSA run has been created, the command-line command to execute a PEATSA run is (all on one line):

```
nohup python path/to/PEATSA.py /path/to/trade_study_file.py > /path/to/peatsa.out 2>&1 &
```

The following is a breakdown of the different elements of this command:

- `nohup`: Ignore the hangup signal so that the process does not stop if the user logs out or is diconnected.
- `python path/to/PEATSA.py /path/to/trade_study_file.py`: Use python to execute PEATSA.py with trade_study_file.py as a command-line argument.
- `> /path/to/peatsa.out 2>&1`: Redirect standard output and standard error to peatsa.out. PEATSA writes output to this file, so the user can monitor the progress of a PEATSA run by examining the contents of this file. The user can also see if a PEATSA run has died unexpectedly by examining the contents of this file.
- `&`: Run in the background.

3.3 PEATSA Outputs

When a PEATSA run is initiated, information is output to a number of places:

- Standard output and standard error messages are written to the file specified at the command line in the PEATSA execution call. (See Section 3.2.)
- Logging information is written to the file specified using the logfile option in the PEATSA options file.
- A new directory is created in the location specified by the run_name and working_directory options in the PEATSA options file.

The contents of the new directory created by PEATSA are as followed:

- docs/. This directory contains two types of files:
  - PEATSAhistory.csv. Only a single file of this type is created, and that file is overwritten at the end of each PEATSA iteration. This file contains extremely basic information about each PEATSA iteration, such as how many EMTG runs were executed, how many EMTG runs found a feasible solution, and how many unique PEATSA cases found a better value of the PEATSA objective function in a given iteration. This information is useful for knowing if a PEATSA run should be stopped, tweaked, and re-started (e.g., if no cases are producing feasible results) or if a PEATSA run should be ended (e.g., because no cases have found more optimal solutions in several iterations).
  - IterationX.csv, where X is the iteration index, starting with 0. A new file of this type is created at the end of each PEATSA iteration. Each file gives information about the most optimal case that has been found up through that iteration for each unique PEATSA case, as defined by the fingerprint. (See Section 3.1.3.) Some information is given by default, such as the file path to the *.emtg and *.emtgopt file, and the feasibility index. Additional information is added by adding tuples to the extra_csv_column_definitions list in the PEATSA options file. (See Section 3.1.3.)
- results/. This directory contains subdirectories in which the output *.emtg, *.emtgopt, and *.emtg_spacecraftopt files produced for each PEATSA iteration are written.
- cases/. This directory contains subdirectories in which the input *.emtgopt files produced for each PEATSA iteration are written.
- images/. This directory contains plots generated by the PEATSA iterations.
- HardwareModels/. This directory contains EMTG hardware files needed for each PEATSA iteration (e.g., .emtg_spacecraftopt).
- PEATSA_iteration_kill_file.DEATH. The contents of the text file may be edited to end a PEATSA iteration or an entire PEATSA run. See the comments in the file for more details.

- PEATSA_MidBake_Options.py. The file is similar to a PEATSA options file, but it specifically only has options that a user is allowed to change from iteration to iteration of a PEATSA run.
- PEATSA_ExecutedOptions.py. This file holds the values of the PEATSA options used to kick off the PEATSA run.

3.4 PEATSA Tips and Tricks

- Use the Linux 'top' command to see what processes are currently active on a server. This can be used to make sure that a PEATSA run is currently executing. This can also be used prior to kicking off a PEATSA run to make sure that someone else is not already using the server.
- Use the Linux 'tail' command to observe live updates to the contents of a logfile or case output.
- The python script **<EMTG_root_dir>**/PyEMTG/PEATSA/grab_best_peatsa_results.py is used to take the full results of a PEATSA run and extract only the feasible case with the best value of the objective function for each unique trade case. This can be a great space-saver because it can allow the user to delete a full PEATSA run without losing the best solutions. If desired, the best results can be used as seed cases for a new PEATSA run to "continue" an old run, too. However, be careful, because this script does not save summary csv files, etc.