GMAT Automation Software User Manual

Table of Contents

No table of contents entries found.

# Introduction

The GMAT Automation Software is a related set of Python scripts used to generate and run batches of GMAT model files. These model files vary in terms of resource values in accordance with an Excel workbook referred to as a “configspec”.

The Python scripts are:

* modelpov.py
* fromconfigsheet.py
* modelgen.py
* gmat\_batcher.py

These scripts will be discussed in the sections below.

## Assumed Directory Structure

“OUTPUT\_PATH” is defined in gmat\_startup\_file.txt located in the bin directory of the installed GMAT application (in Windows this is under the %APPDATALOCAL% defined directory.)

GMAT uses this directory to store or load model files.

This system is designed with the assumption that there is a “Batch” directory and a “Report” directory located under the OUTPUT\_PATH reference, e.g. “[OUTPUT\_PATH]/Batch/” and “[OUTPUT\_PATH]/Report/”.

## System

Use of Excel as the source of configspec constrains this system to execution on the Windows platform. The environment variable, %APPDATALOCAL%, was brought into usage on Windows 7 and subsequent.

Execution of GMAT Model Files is math intensive and requires significant platform resources to execute. The gmat\_batcher script executes model files in GMAT command line mode, under a concurrency model aimed at maximizing the parallel execution of models. A time-out value of 10 minutes (600 seconds) is used to avoid “hung” threads. If the platform is unable to execute a model within this time, the system will not allow it to complete.

The gmat\_batcher.py execution is benchmarked with an HP-Z600 workstation as follows:

* Dual Intel® Xeon® E5506 CPUs at 2.13GHz
* 16GB of Installed DDR3 Ram
* 64-bit Windows 10 Pro, version 1803
* NVIDIA GTX 750i GPU

# GMAT Model Files

The topology of the GMAT model files used in this automation scheme are part of the design.

ModelMissionTemplate.script is the top-level model definition in the batch system. It will be copied and renamed for each unique instance of a mission by modelgen.py.

The mission and spacecraft configuration is read from an Excel workbook by modelgen.py, which creates and saves a mission specific include file in the Batch directory. Each generated include file is uniquely named with configuration, epoch, Julian day and time of generation, and each filename is prefixed with the string "Include\_".

The modelgen.py script appends an #Include macro referencing the created include file to the copy of ModelMissionTemplate.script. Two other invariable #Include macros are also appended to ModelMissionTemplate.script:

1. Include\_MissionDefinitions.script
2. Include\_StaticDefinitions.script

The appended copy of ModelMissionTemplate.script is saved in the Batch directory. It is important that ModelMissionTemplate.script be kept small, because modelgen.py may copy it hundreds of times. This is why the bulk of the Resource definitions are factored into the invariable macro Include\_StaticDefinitions.script.

The invariable macros are not edited by modelgen.py, they must be maintained by the user.

ModelMissionTemplate.script and its include files are intended to be executed from the GMAT command line. The GMAT GUI cannot be used to edit these files. Instead, create a new model, proceed with design and test using the GMAT GUI.

The model created (or changed) in the GUI should be factored and copied from the GUI generated model file to the batch template files located in the GMAT "OUTPUT\_PATH":

1. Include\_MissionDefinitions.script
2. Include\_StaticDefinitions.script
3. ModelMissionTemplate.script

Update the configuration workbook any time spacecraft or its hardware is changed, likewise the ModelMissionTemplate.script. Rerun modelgen.py following update of the workbook.

# Excel “ConfigSpec” Workbook

Interface Agreement: an Excel workbook exists which contains a sheet named "GMAT" having a contiguous table starting in cell "A1". The table consists of a first line of parameter names and successive lines of spacecraft properties and relevant hardware configuration.

The first line table headings are usually not the same as GMAT resource names. The associated routine, "modelpov.py" defines a mapping of required GMAT resource names to worksheet table headings, which we refer to as parameter names.

Procedure “modelgen.py” will use this association to generate the correct GMAT resource names.

Procedure “fromconfigsheet.py” will extract only the parameter names defined in modelpov.py. Note that it is possible with this logic to extract NO data from the workbook, in this case the “modelpov.py” file may be edited to include the intended parameter names, or the workbook may be so edited.

Figure 1 shows an example ConfigSpec generated from analysis data using an Excel SQL query.

|  |
| --- |
|  |

Figure , GMAT ConfigSpec Table in Excel Workbook

Variation of orbital elements is independent of hardware configuration. Specifically, inclination cases may be multiple for each given "GMAT" table row and are gleaned from a separate n x 1 table of values in named range, "Inclinations" contained in a sheet named "Mission\_Params".

|  |
| --- |
|  |

Figure , Mission Parameters in ConfigSpec Workbook, Starting\_Epoch Highlighted

Figure 2 shows an example "Mission Params". Note that there are more orbital elements on than are presently implemented.

At the top right in cell B1, the mission name must be defined in a named range called “Mission\_Name”. The mission name is used by the script to name various files and can be any string by which the user can identify model files associated with this set of configurations and mission definitions.

Cases of initial epoch are defined by an n x 4 array of values in named range, "Starting Epoch". Each row (n) contains a UTC formatted time and date value in column 1, e.g. "20 Mar 2020 03:49:00.000 UTC".

For display purposes columns (n,2), (n,3), and (n,4) contain a GMAT viewpoint vector consisting of x, y, and z components of rendering camera position (J2000 ECI coordinate system). These viewpoint vectors are associated with each epoch value. This is under the assumption that the GMAT application is executed with the GUI enabled, which will be a performance burden.

# Points of Variation

The “modelpov.py” script encapsulates the Points of Variation in a Python dictionary structure for GMAT model generation. It is used by "fromconfigsheet.py" and "modelgen.py”.

The implemented points of variation in the model file are listed here. The list form is:

Table Heading: [top model name] [sub-model name].[resource name].

Points of variation:

* Dry mass: Spacecraft EOTV.DryMass
* Starting Epoch: Spacecraft EOTV.Epoch
* This is a list of epoch values to be executed.
* Inclination: Spacecraft EOTV.INC
* This is a list of inclination values to be executed.
* Costate: The constraint value for final SMA and Inclination change
* This is not actually a model parameter but is used to select a JSON control file.
* Max Thrust Power: ElectricThruster HET1.MaximumUsablePower
* Min Thrust Power: ElectricThruster HET1.MinimumUsablePower
* Efficiency: ElectricThruster HET1.FixedEfficiency
* Isp: ElectricThruster HET1.Isp
* Thrust: ElectricThruster HET1.ConstantThrust
* Available Power: SolarPowerSystem EOTVSolarArrays.InitialMaxPower
* Propellant: ElectricTank RAPTank1.FuelMass
* Output: ReportFile1 ReportFile1.Filename
* Viewpoint: Orbit View DefaultOrbitView.ViewPointVector

In order to cover the range of eclipse conditions the EOTV Epoch is typically varied for the four seasons:

* 20 Mar 2020 03:49 UTC
* 20 Jun 2020 21:43 UTC
* 22 Sep 2020 13:30 UTC
* 21 Dec 2020 10:02 UTC

The epoch dates are specified in the configsheet workbook named as:

* Epoch\_1
* Epoch\_2
* Epoch\_3
* Epoch\_4

The Viewpoint is superfluous in most cases, since model execution is intended to be in batch mode for this system. However, if graphic output is desired, the OrbitView viewpoint is varied associated with the Starting Epoch as follows:

* 20 Mar 2020 03:49 UTC, (80000, 0, 20000)
* 20 Jun 2020 21:43 UTC, (0, 80000, 20000)
* 22 Sep 2020 13:30 UTC, (0, -80000, 20000)
* 21 Dec 2020 10:02 UTC, (-80000, 0, 20000)

The ReportFile1.Filename and the Model name are generated by concatenating the configuration, the Starting Epoch and the Inclination.

# Generation of the Batch Files

Script “modelgen.py” produces GMAT model Include files containing variants of model resource values and parameters. The code supports batch processing of different mission scenarios in which the spacecraft configuration and/or initial orbital elements vary.

The approach utilizes the GMAT 2018a #Include macro, which loads resources and script snippets from external files. The script creates include files whose parameter and resource values vary in accordance with the configspec Excel workbook.

A top level GMAT script template must exist in the GMAT model directory. This script template will be copied and modified by modelgen.py. The script template must contain the GMAT create statements for GMAT resources.

The script template will be appended by three #Include statements referencing include files as follows:

1. a GMAT script file defining the static resources, those that don't change per run.
2. An include file generated by "modelgen.py".
3. An include file containing the GMAT Mission Sequence definition, which doesn’t change per run.

The appended script template will be copied to a "batch" directory under a mission specific filename. An example is:

AlfanoXfer\_64HET8060W\_16000.0kg\_20Jun2020\_28.5\_J039\_172018.script

which references:

Include\_16HET8060W\_16000.0kg\_20Jun2020\_28.5\_J039\_172018.script

There will be a unique copy of the script template for each line of the configspec workbook, elaborated by the number of inclination cases, and elaborated by the number of Starting Epoch cases. The total number of files, template and include, will be:

2 \* [Number of configspec rows] \* [Number of inclination cases] \* [Number of Starting Epoch cases]

Once these files are copied, a list of their filenames will be written out as contents of a batchfile in the GMAT model OUTPUT\_FILE defined path. The model name will be of the form:

[Mission Name] + '\_\_RunList\_[Julian Date-time] + '.batch'

An example is:

"AlfanoXfer\_\_RunList\_J009\_0537.25.batch".

A dictionary is used to drive the actual resources and parameters written to Include macro files.

The dictionary is factored into "modelpov.py" such that additional resources may be added or deleted with minimal change to code in “modelgen.py” and “fromconfigsheet.py”.

Include file 1 is an invariable file copied from from the initial model file written by GMAT. This is a large file of Resources and Parameters which do not change from run-to-run.

In the case of a new model or change …

The points of variation probably should be updated in "modelpov.py" a new model is generated.

The external module "fromconfigsheet.py" is called to read excel worksheet to update the values of the dictionary.

The Alfano trajectory is used in the current model mission, and a user defined parameter, the costate, must be updated in concert with the inclination to execute the Alfano-Edelbaum yaw control law. The costate calculation is out of scope as of this version [TODO]. The costate is specified on the Mission\_Params sheet of the workbook.

Notes:

1. To model the return trip of the reusable vehicle, two include files must be generated, one with payload mass included, one without. This should be handled in the configspec workbook if a return trip is required.
2. Dry mass varies with the vehicle power and thrust.
3. Efficiency, thrust and Isp vary with the selected thruster set-points.
4. In order to cover the range of eclipse conditions the EOTV Epoch is varied for the four seasons:

20 Mar 2020 03:49 UTC

20 Jun 2020 21:43 UTC

22 Sep 2020 13:30 UTC

21 Dec 2020 10:02 UTC

The epoch date is specified in the configsheet workbook.

1. Propellant is given as an initial calculation in the workbook, then the actual value from a run is substituted and the model rerun until convergence. This is presently a manual process.
2. A GMAT Report File is output into the OUTPUT\_PATH/Report directory for each run. This is a .csv file and its name must be unique for each iteration of the model. The ReportFile1.Filename is generated by “modelgen.py” which suffixes the Julian date-time to the filename.
3. The OrbitView viewpoint is superfluous in most cases, since model execution is intended to be in batch mode for this system. However, if graphic output is desired, the convention is to vary the viewpoint with the Starting Epoch
4. The top level GMAT script is be called by the batch facility, “gmat\_batcher.py”.
5. TODO: the initial baseline depends on the exact spacecraft and hardware created in the top-level template. Five of these represent points of variation:

* Create Spacecraft EOTV;
* Create ElectricThruster HET1;
* Create SolarPowerSystem EOTVSolarArrays;
* Create ElectricTank RAPTank1;
* Create ReportFile, multiple files possible, various names

These instance dependencies can be avoided by reading and interpreting ModelMissionTemplate.script where the user will copy his definitions of these resources.