GMAT User-Defined Function Requirements

Submitter	Owner	Status	Last Updated	Targeted Implementation
S. Hughes	E. Dove	Formulation	3/18/2008	Apr. 08

Feature description and Scope

The function feature allows the user to define custom functions. User defined functions are useful for isolating repetitive calculations that must be performed in the mission sequence. GMAT shall allow a user to create functions in a general way by defining a list of input arguments, a series of operations and events to be performed on the input arguments, and a list of output arguments to be passed back to the main mission sequence or calling function.

One way to think about user defined GMAT functions is to consider them as mini mission sequences with their own set of resources and events. The idea is to isolate small mission analysis problems, such as Lambert's problem, into mini-missions that GMAT can solve as a separate process, and then bring the desired output back into the main mission sequence or calling function.

The input and output arguments to a user function can be variables, arrays, strings, and GMAT resources. Inside a user-defined function the user can create and define local variables, arrays, strings, and GMAT resources. All input and locally created items can be used in mathematical expressions as well as events such as Propagate, Optimize, control flow, etc. Changes to the input arguments are local only to the user-function unless the input argument is also contained in the output argument list. All user-defined functions can be called from within a user defined function, including a function calling itself.

Rationale and Feature Justification

The function feature is useful to all users of GMAT and all GMAT applications. Functions reduce the duplication of code in a program, enable reuse of code across multiple programs, decompose complex problems into simpler pieces thereby improving maintainability and ease of extension, improve readability of a program, and facilitate information hiding .[1]

Assumptions

- All global objects, including Solar System and path info, are reset between runs.
- All numbers in GMAT are treated as real.
- Matlab interfaces are treated as global objects. Matlab interfaces do not require recreation or reconfiguring inside of functions.
- The behavior of MATLAB® calls does not change.

Definitions

- **CallFunction** is the name of the command used in GMAT to call all function types available to a user.
- **"File Open" panel** is the file selection window that is used to select the file of your choice by browsing to the directory of your choice.

[1] Advantages extracted from Wikipedia's Function (computer science) article.

- **Function Control Sequence** is the sequence of commands built from a function
- **Function** in the context of this requirements document, the word function pertains to user-defined GMAT functions unless stated otherwise.
- **Global objects** exist at the scope of the mission sequence. If GMAT is extended to support multiple mission sequences, there will be a global store for each mission sequence.
- **Literals** is a notation for representing an "as is" value. For our application of the word, literals are using numbers and strings, like 2 and 'MyString' respectively, for function inputs, instead of assigning numbers and strings to variables and using them as function inputs.
- **Mission Control Sequence** is the sequence of commands built from the main script.
- **Path** is the folder location in either relative or absolute notation excluding the filename.
- **Validation** of the contents of a function consists of checking the syntax for violations to the GMAT scripting environment and checking for matching object types passed in and out of the function call.

Functional Requirements

1. Core
   - 1.1. The system must allow the user to employ any command supported by GMAT, inside of functions.
   - 1.2. The system must allow functions to be called inside of functions and the main body of a script.
   - 1.3. The system must allow function recursion, the ability for a function to call itself.
   - 1.4. The system must allow function iteration, the ability for a current function to call another function and for that function to call the current function.
   - 1.5. The system must allow MATLAB® to be called from inside of functions. A MATLAB® function can not call a GMAT function.
   - 1.6. The system must only support functions with contents residing inside of a .gmf ascii file.
   - 1.7. The system must perform validation on all declared functions when GMAT performs the following actions:
     Run, Build, Build&Run
   - 1.8. The system must reset the function control sequence to the current contents of the function file at runtime.
   - 1.9. The system must only allow one function to be defined inside of a function file. Two or more functions can not exist in one GMAT function file. If more than one function is present in a file, a warning will be thrown and the system must only accept the first function in the file.
   - 1.10. The system must process user actions to pause or stop while processing a function.
   - 1.11. The system must require that a function is declared before use. Function declaration consists of GMAT parsing through CallFunction lines and collecting a listing of all functions that are not Predefined, Matlab, or other non-GMAT user-defined functions. The collected listing contains the declared user-defined functions. The use of a Create line is not needed for user-defined function.
   - 1.12. The system must only allow the use of functions that are found on user specified function paths. Multiple paths can be used.
   - 1.13. The system must allow the adding and removing of GMAT function paths in the script. Adding function paths in the script can only occur in the Resource level of the main script.
   - 1.14. The system must order multiple paths based on the ordering the path was added. If there are multiple GMAT functions of the same name in the paths specified, the function used is the one from the most recently added function path.
2. Scoping

2.1. The system must allow users to create local variables and objects inside of functions. Any type of object that can be created and used in the main mission sequence must be creatable and useable inside of a function.
2.2. The system should treat the GMAT solar system model as a global parameter. Solar System objects do not need to use the Global syntax to be set as global.
2.3. The system must treat Coordinate Systems as global. Coordinate System objects do not need to use the Global syntax to be set as global.
2.4. The system must treat Propagators as global. Propagator objects do not need to use the Global syntax to be set as global.
2.5. The system must allow users to declare any previously created object global, excluding the objects that are always global, in all instances it is used. Objects that are declared global are available in the main script, and internally in functions without being explicitly stated in the input sequence.
2.6. The system must declare global objects in the scope of each function or script that use the object, but only create the object in one of these elements. Global objects are declared using the "Global" keyword, which functions the same way MATLAB®'s "global" keyword functions.

3. Input/Output
3.1. The system must support the following as function inputs:
3.1.1. Entire objects
3.1.2. Object properties of numeric or string type
3.1.3. Real number variables and arrays
3.1.4. String variables
3.1.5. Literals
3.2. The system must support the following as function outputs:
3.2.1. Entire objects
3.2.2. Object properties of numeric or string type
3.2.3. Real number variables and arrays
3.2.4. String variables
3.3. The system must support calling a function with no inputs and/or outputs.
3.4. The system must, upon return from a function, not change non-global function inputs unless they also appear in the output argument list.
3.5. The system does not require a user to use a Create command inside of a function for the inputs passed through the function.
3.6. The system must require the user to use a Create command inside of functions for the output parameters that are not also input parameters. If an output parameter is global and created outside of the current function, no Create command is necessary inside the function, only a Global command.
3.7. The system must require a user to use square braces for the output parameters of a function call, even if there is only one.
3.8. The system must allow the use of subscribers inside functions.
Refer to the *Subscriber Functionality in Functions Requirements* for additional details.

Script Requirements

The syntax for declaring functions in a script corresponds to the function path. All filenames with a valid GMAT Function extension can be a declared GMAT function. Refer to *Functional Requirement* 1.11 for additional details on function declaration requirements. Refer to *Functional Requirement* 1.12- 1.14 for additional details on function path requirements. Function paths can be added through the startup file and inside of the script. Function paths can only be removed from inside of the script.

Examples of adding and removing function paths are presented below:

From the startup file and script:
```
GmatFunctionPath.Add = ./GmatFunctions/
GmatFunctionPath.Add = C:/GMAT/functions/GmatFunctions/
```
or
From the script only:
```
GmatFunctionPath.Remove = C:/GMAT/functions/GmatFunctions/
```

The syntax for function calls, including how the user specifies inputs and outputs, must be identical to MATLAB's® syntax for function calls.

Examples of calling a function are presented below:

```
[Out1, Out2,....., OutM] = MyGMATFunction(In1, In2, ...  InN)
```
or
```
MyGMATFunction(In1, In2, ...  InN)
```
or
```
[Out1, Out2,....., OutM] = MyGMATFunction
```
or
```
[Out1] = MyGMATFunction
```
or
```
MyGMATFunction
```

The syntax for the first uncommented line in the contents of a function, must be consistent with MATLAB's® syntax for the first uncommented line of their function contents.

Examples of the first uncommented line of a function's contents are presented below:

```
function [Out1, Out2,....., OutM] = MyGMATFunction(In1, In2, ...  InN)
```
or
```
function MyGMATFunction(In1, In2, ...  InN)
```
or
```
function [Out1, Out2,....., OutM] = MyGMATFunction
```
or
```
function [Out1] = MyGMATFunction
```
or
```
function MyGMATFunction
```

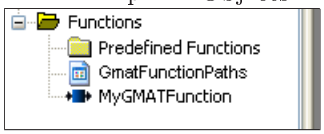
GUI Requirements

1. The system must allow the GUI to emulate function functionality available from the script.
2. The system must only allow users to add functions from the CallFunction properties page.
3. The system must have a Global checkbox on every object's properties panel that is capably of being Global. Objects that are always global should have an un-checkable Global check box.
4. The system will not show global objects declared inside of a function in the GUI. For example, if a variable MyVar was created in a function and marked as global and used in the main script, MyVar would not show up in the GUI.

GMAT function folders and files

The functions declared, based on *Functional Requirement* 1.11, are populated in the Resources Tree under the Function folder. Figure 1 illustrates the types of Resource Tree files a user would encounter related to GMAT functions, excluding the Predefined Functions folder.

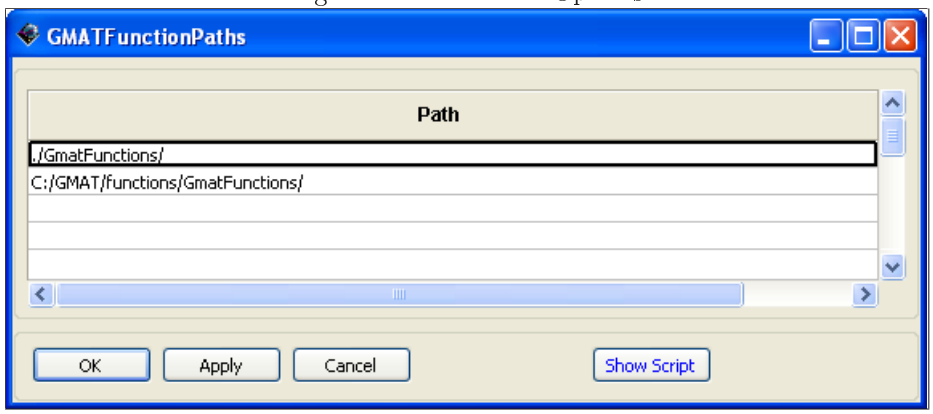
Figure 1: Example of Objects in Tree



Changing Function Path

Upon opening the properties panel for the GmatFunctionPaths object in the Resource Tree, the user has the ability to change the path information. Figure 2 illustrates how a user can edit the function paths that are currently used.

Figure 2: Path Folder Options



1. Path grid - Accepts relative or absolute path information. The path on top is the first path used.
   1.1. Right clicking a cell brings up a properties panel relevant to the cell, similar to the Propagate and Control Flow grid cells. The Path cells open the "File Open" panel in the selected function path folder. The "File Open" panel only allows files of a type specified in *Functional Requirement* 1.6 to be selected.

Global Object Check Box

Objects at the Resource level all need a "Global" checkbox, which is initially unchecked except for the objects that are automatically global. The Global check box for objects that are always global are marked as checked and uneditable. Figures 3- 4 illustrates examples of the Global object check boxes.

Figure 3: Global Check Box (Editable)

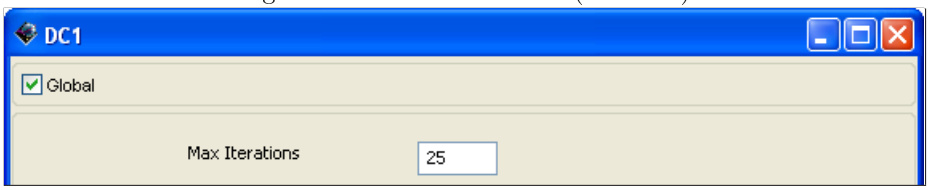
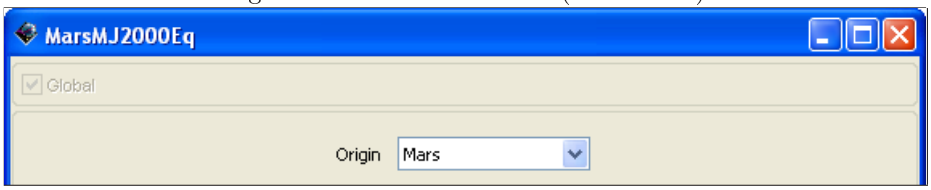


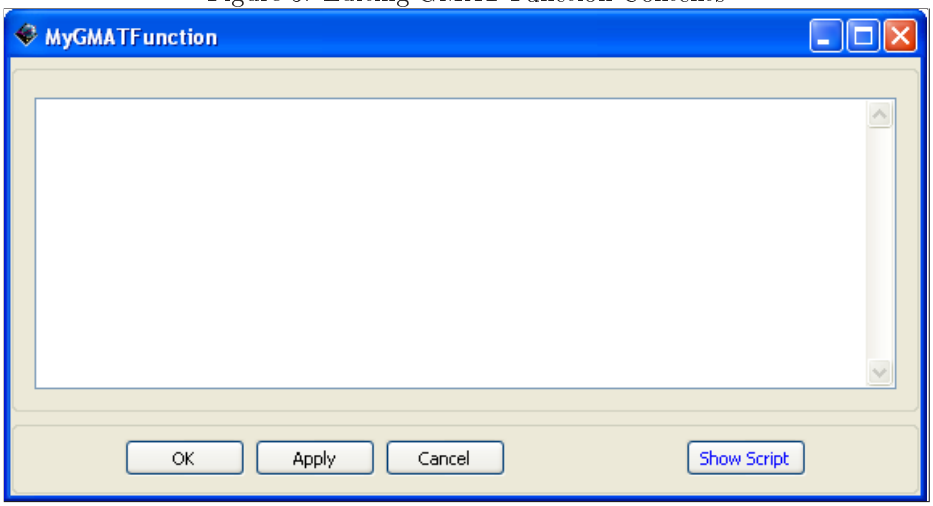
Figure 4: Global Check Box (Uneditable)



Editing Function Contents

A user can not create a new GMAT Function file from the GUI but they can load the contents of an existing declared function and edit its contents. Figure 5 illustrates the interface for a user to edit the contents of a GMAT Function.

Figure 5: Editing GMAT Function Contents



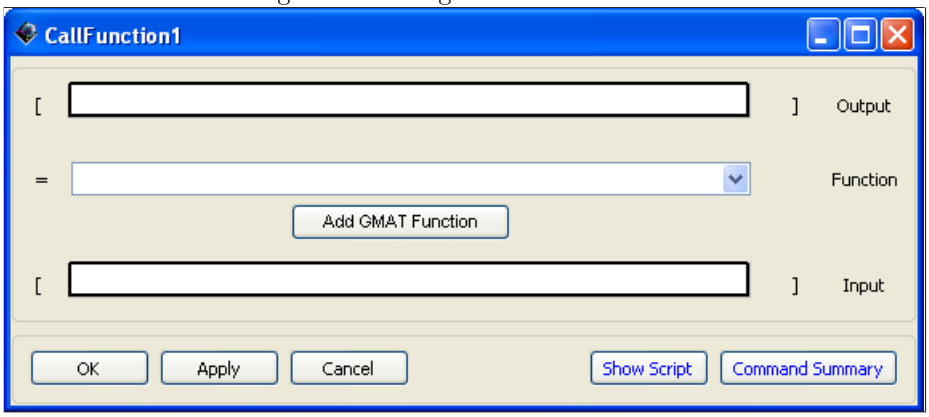
1. Function contents text box - Contains the contents of the function specified in the File text box.
   1.1. The shortcut keys available in the script editor are available here.

2. The functionality of the OK, Apply, Cancel, and Show Script buttons are consistent with the rest of the system.

Calling a GMAT Function

On the GUI, the CallFunction's properties panel is the only area a user can add or select a function for use in the mission sequence. Figure 6 illustrates a CallFunction panel to allow a user to take advantage of the added capabilities presented in this document.

Figure 6: Calling a GMAT Function



1. Input/Output grid - Functionality remains the same.
2. Function Drop Down - The contents of the drop down are populated by MatlabFunctions and previously declared GmatFunctions, similar to the way the Resource tree will display all usable functions.
3. Add GMAT Function button - Upon activation, opens a "File Open" panel set to the first directory in the GmatFunction path and is populated by the file extensions listed in *Functional Requirement* 1.6. The user selects the GmatFuction from those displayed, or by navigating to a different folder.
4. The functionality of the OK, Apply, Cancel, Show Script, and Command Summary buttons are consistent with the rest of the system.

Implications for Other Features

1. The mission tree should only show one level of function calls. If a function call is made inside of a function call, the nested function call will not appear in the mission tree. In this event, the user can see the internal function call by opening the function file and reading its contents.

Considerations for Future Enhancements

1. The design must not preclude the ability to output errors from functions with incorrect syntax in a similar way errors are outputted from the main script. The errors should include the line with line number in the main script where the function is called and the line with line number in the function where the incorrect syntax is located.
2. The design must not preclude the GUI ability to order function paths once they are added.
3. The design must not preclude the ability to obtain a resource object summary that would include details of all objects, including whether an object is global.
4. The design must not preclude the ability to add functionality to the CallFunction syntax that is similar to MATLAB®'s nargin and nargout feature.
5. The design must not preclude the ability to call a function that returns exactly one value (variable, array element, or object parameter) in an inline math assignment command. For example, for a user created GMAT function named func, which produces one variable as its output, the following syntax is acceptable:

```
varOut = 1 + 3*(func(varIn1) + func(VarIn2));
```

Unresolved Issues and Requirements

1. The Plot command is not created yet.

Tests and Use Cases

1. I/O Tests
2. Internal Computation Tests
3. Solar system tests
4. MATLAB® call tests
5. Variables created inside a function are only available inside a function
6. Output tests
   6.1. XYPlot
   6.2. OpenGL Plot
   6.3. ReportFile