GMAT Function Requirements

Submitter	Owner	Status	Last Updated	Targeted Implementation
S. Hughes	E. Dove	Formulation	11/7/2007	Mar. 08

Feature description and Scope

The function feature allows the user to define custom functions (subroutines). 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.

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.

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 objects. All input and locally created items can be used in mathematical expressions as well as events such as Propagate, Solve, 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.

Rational 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.

Assumptions

None.

Functional Requirements

1. The system must support the following as function inputs:
   - 1.1. Entire objects
   - 1.2. Object properties
   - 1.3. Real number variables
     (There is no distinction between real and int in GMAT)
   - 1.4. String variables
   - 1.5. Literals
     (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.)
2. The system must support the following as function outputs:
   - 2.1. Entire objects
   - 2.2. Object properties
   - 2.3. Real number variables
   - 2.4. String variables

3. The system must allow users to create local variables and objects inside of functions. Any object that can be created and used in the main mission sequence must be creatable and useable inside of a function.
4. The system must allow the user to use GMAT's inline math capability inside of GMAT functions.
5. The system must support calling a function with no inputs and/or outputs.
6. The system must allow the user to employ any command supported by GMAT, inside of functions.
7. The system should treat the GMAT solar system model as a global parameter. The solar system is not required as a function input, and settings on the solar system outside of functions will affect the solar system values inside of functions.
8. The system must allow functions to be called inside of functions and the main body of a script.
9. The system must allow function recursion, the ability for a function to call itself.
10. 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.
11. The system must, upon return from a function, not change function inputs unless they also appear in the output argument list.
12. The system must allow MATLAB® to be called from inside of functions.
13. The system must support global objects. Objects that are declared global are available in the main script, and internally in functions without being explicitly stated in the input sequence.
14. The system must treat Matlab interfaces as global objects. Interfaces do not require recreation or reconfiguring inside of functions.
15. The system must allow global objects to be created inside of functions.
16. The system must treat Subscribers as global in the sense they can be written to using the Report or Plot commands from within functions, even if the Subscriber was not included in the function calling sequence.
    [UNRESOLVED] Subscribers will not update inside functions unless a Plot or Report command is issued.
17. The system must treat Coordinate Systems as global.
18. The system must treat Propagators as global.
19. The system must only support functions with contents residing inside of a .m or .gmf ascii file.
20. The system will use a default GMAT function path based on the FUNCTION_PATH defined in the GMAT startup file.
21. The system must require that a function is declared before use. A function declaration can occur inside of the main script or function. A function is declared once GMAT stores the path and filename of the function as a GMAT Function object.
22. The system will preform a syntax check on all declared functions when building a script.
    [UNRESOLVED] Does that include checking globals and parameters for the right types?
23. The system will only allow one function to be defined inside of a a function file. Two or more functions can not exist in one GMAT function file.
24. The system must process user actions to pause or stop while processing a function.
25. The system must declare global objects in the scope of each function or script that use the object, but only created in one of these elements. Global objects are declared using the "Global" keyword, which functions the same way MATLAB's "global" keyword functions.
26. The system must have global objects exist at the scope of the Sandbox. Global objects are not shared between separate Sandboxes.
27. The system must have global objects that are not part of the configuration or solar system created during Sandbox initialization.

Script Requirements

The syntax for declaring a function in a script contains a function file of the same name and a function path, with the function path being optional. If no function path is designated the default function path and

the declared function name is used.

Examples of declaring a function from a script are presented below:

```
Create GmatFunction MyGMATFunction;
   GMAT MyGMATFunction.FunctionPath = ./functions/MyGMATFunction.gmf;
   or
   Create GmatFunction MyGMATFunction;
```

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
   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 MyGMATFunction
```

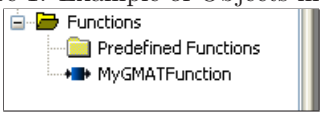
GUI Requirements

1. GMAT Function functionality available in the script will be available from the GUI.

GMAT function Resource Tree objects

Figure 1 illustrates the types of Resource Tree objects a user would encounter related to GMAT functions, excluding the Predefined Functions folder.

Figure 1: Example of Objects in Tree



Adding a GMAT function

The user can only add a GMAT Function by right clicking the Resource Tree Function folder and selecting Add -> GMAT Function Path. Figure 2 illustrates how a user can add a GMAT Function to GMAT.

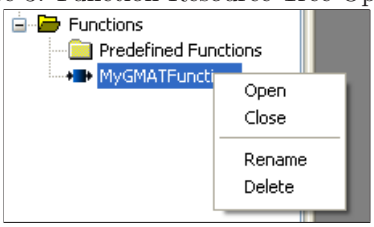
Figure 2: Adding Function

1. Selecting GMAT Function option brings up the panel shown in Figure 3

Tree options for GMAT function object

In order to delete or edit the function, a user would have to right click the function object and select from the list of options(Open, Close, Rename, and Delete). Figure 3 illustrates the options a user can have when right clicking the folder that contains GMAT functions in a specified path.

Figure 3: Function Resource Tree Options



1. The functionality of the Open, Close, and Delete options must be consistent with the rest of the system.

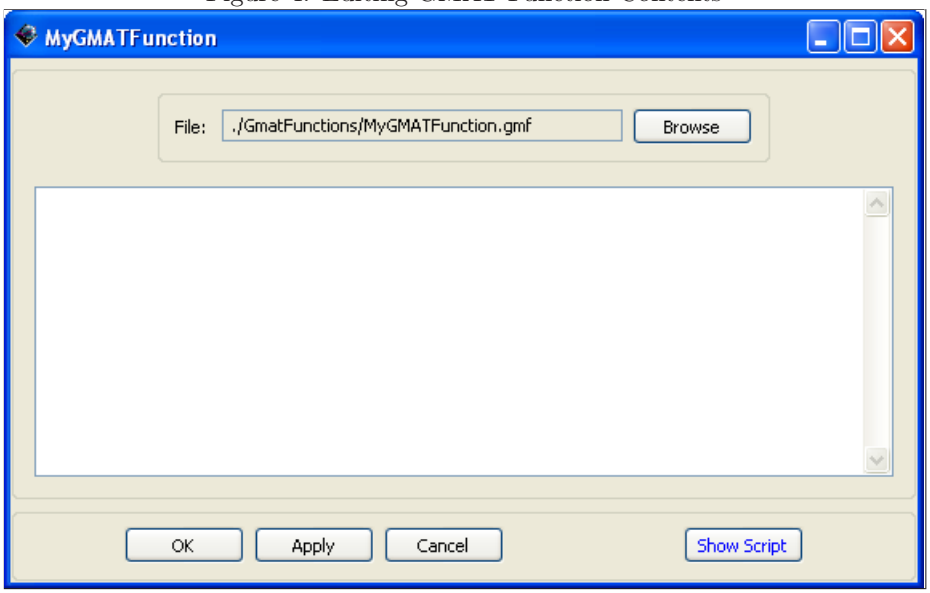
2. Rename - Renames the stored GMAT Function name and the GMAT function filename.

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 function in the GMAT Function path and edit the contents. Figure 4 illustrates how a user can edit the contents of a GMAT Function.

1. File text box - Displays relative or absolute path information. This input field can not edited directly. Its contents can only be updated by using the Browse button or from reading in a script.
2. Browse button - Upon activation, opens a "File Open" window.
   - The "File Open" panel opens in the selected function path folder.
   - The "File Open" panel only allows files of a type specified in this requirements document to be selected.
   - This button is always selectable.
3. Function contents text box - Contains the contents of the function specified in the File text box.
   - The shortcut keys available in the script editor are available here.
4. The functionality of the OK, Apply, Cancel, and Show Script button is consistent with the rest of the system.

Figure 4: Editing GMAT Function Contents



Implications for Other Features

1. The mission tree should only show one level of function call. 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 ability to use Include instead of Create to declare functions.

Unresolved Issues and Requirements

1. The Plot command must be created in order to allow OpenGL and XYPlot output capabilities from inside a function.

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