GMAT User-Defined Function Requirements

Submitter	Owner	Status	Last Updated	Targeted Implementation
S. Hughes	E. Dove	Formulation	11/13/2007	Mar. 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.[Add Wikipedia reference]

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**
- **Function Control Sequence**
- **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.
- **Mission Control Sequence**
- **Path**
- **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
      (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.)
   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 and outputs passed through the function.
   3.6. The system must require a user to use square braces for the output parameters of a function call, even if there is only one.

4. Unresolved
   4.1. 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.

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

THIS SECTION IS BEING UPDATED

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 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.

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