Comparison of Options for GMAT's C-Interface

Darrel J. Conway
Thinking Systems, Inc.

January 12, 2012

Abstract

This document compares approaches to building GMAT's C-Interface.

1 Introduction

Whatever. The goal here is the table below.

2 Notes

Below there is a table stating what I know so far. This is basically notes while looking at the options. Other notes:

- Another group looking to use C++ in MATLAB and Python produced this report: http://verdandi.gforge.inria.fr/doc/high_level_interface.pdf
- This URL looks interesting and pertinent: http://undocumentedmatlab.com/

Open questions/comments:

- Is this project misnamed? Most of the approaches we discuss are really Java interface issues, not C interface. It seems to me that C only enters as an intermediary. (I think we do want a C interface. But if the goal is really to talk to MATLAB, Java seems the most natural approach.)
- Java calls in MATLAB are straightforward. You add path data to the MATLAB Java path, and then call into the .jar or .class files. JNI/JNA calls are a bit more convoluted because MATLAB needs to be able to find the associated shared libraries.
- For what it's worth, I'm becoming more and more convinced that we do want Python support along with MATLAB support. That seems to point to a native C interface to me, and postprocessing (swig or something like it) to get the further hooks.

Tool	URL	Notes
loadlibrary		• Method prototyped last year for propagation proof of principle • Requires C wrappers for GMAT classes • Interface changes inside of GMAT will ripple through the interface code, and need hand coding to adapt
MEX	www.mathworks.com	• Requires specific MEX function access to each exposed piece of code. • Interface changes inside of GMAT will ripple through the MEXFunction code, and need hand coding to adapt
JNI	docs.oracle.com/javase/7/docs/ technotes/guides/jni/index.html	• Supports classes • Has the reputation of being cumbersome to use and error prone.
swig	www.swig.org	• There is a lot of chatter on SWIG-MATLAB interconnections, but not a lot of information about how well/if it works. One interesting project is SwigMatlabPlus • SwigMatlabPlus doesn't seem to be available anymore – or will take some work to track down. The link (http://alumni.media.mit.edu/ ˜sbasu/code/swigmatlabplus/) does not include the source, and other links on the developer's web site at MIT are broken. It is Windows/Visual C++ specific. • Requires a custom .i file to specify what the interface exposes • swig builds JNI files and java files used to interface into Java, along with (at least one) .c file to link into the shared library. • A note from the swig manual: "If you are going to use optimisations turned on with gcc (for example -O2), ensure you also compile with -fno-strict-aliasing. The GCC optimisations have become more aggressive from gcc-4.0 onwards and will result in code that fails with strict aliasing optimisations turned on. See the C/C++ to Java typemaps section for more details."

Tool	**URL**	**Notes**
JNA	jna.java.net/	• Provides C access, so no direct class support. • Allows for – and requires – an independent API definition through Java classes
BridJ / JNAerator	//code.google.com/p/bridj/	• Appears to be pretty young. Will it survive? • Notes from the website: **Key features** – Dynamic C / C++ / COM interop : call C++ methods, create C++ objects (and subclass C++ classes from Java !) – You never need to compile any native code : we deal with the cross-compilation hassle for you once and for all in BridJ ! (works on Windows, Linux, MacOS X, Solaris, Android...) – Full JNAerator support : stay away from C / C++ headers ! – Small library size (600 kB all included) – Straightforward type mappings with good use of generics • Untested
.NET		• Basically windows specific • May be feasible via mono (www.mono-project.com); downloaded and installed on Linux in AZ, but so far untested.
COM		• Windows specific • COM interfaces are a bit old school, and have been replaced in large part by .NET.
TCP/IP		