JMatlab/Link User Guide

JStatCom Engineering, www.jstatcom.com

Markus Krätzig, March 14, 2007

Contents

1 Introduction

JMatlab/Link is an add-on to the Open Source framework JStatCom.[1] It can be used to call dlls that have been generated with the Matlab Compiler directly from Java. This allows to integrate Matlab features into Java applications and thus to combine the strengths of both.

2 Features

With the Matlab Compiler it is possible to generate stand-alone dlls from typical Matlab .m files easily with a single command, for example:

```
mcc -B cpplib:matlabtest inttest.m graphtest.m
```

Using JMatlab/Link together with the generated dlls has the following benefits:

- works with Matlab Compiler version 4 (R14SP1 and upwards)

[1] JMatlab/Link is published under the GPL, JStatCom under the LGPL.

- full support for Matlab graphics, they will appear in the usual figure windows with all features, for example rotating for 3D graphs and various export options
- allows full control over Matlab execution, especially stopping a running computation any time
- reports all Matlab output and errors to a logging handler (default or user supplied)
- even if a Matlab computation results in a fatal error, the Java application will remain stable and can safely invoke other computation based on Matlab
- allows for a single installation procedure of applications using it without the need to install the Matlab Component Runtime in a separate step

3 Limitations

JMatlab/Link currently only runs on Windows. Currently only the following types can be passed between Java and Matlab:

- scalars, except UInt32
- number arrays, except UInt32
- strings
- string arrays

Internally on the Matlab side there are no such limitations and all Matlab types may be used. Other types, like for example dates, may be coded/decoded as number arrays.

Matlab calls can only be run in a single thread, however this can be a distinct executor thread. JStatCom provides a convenient interface to that. This makes it possible to let the GUI run in a different thread than the calculations, thus keeping it responsive and even allowing the user to stop a running calculation.

The first Matlab call requires an initialization of the runtime, which takes about as long as starting Matlab on your computer. Subsequent calls do not have the initialization overhead.

4 Software Requirements

To compile Matlab dlls, a Matlab 7 installation together with the Matlab Compiler Toolkit is required. Furthermore, a C++ Compiler is needed to generate the dlls. The Lcc compiler that is shipped with Matlab is not sufficient, instead the Microsoft Visual C++ compiler is recommended. Other compilers have not been tested but might work as well.

For the Java part, the framework JStatCom is needed, which consists of a file `jstatcom.jar` and a few 3rd party libraries. It runs with Java 5 and cannot be used with Java 1.4 versions.

JMatlab/Link is a directory that contains the required dlls that enable the communication between Java and Matlab.

A Java IDE, like for example Eclipse, is strongly recommended.

- OS Windows 2000/XP, also runs on Windows Server 2003
- Matlab version 7
- Matlab Compiler version 4
- Microsoft Visual C++ compiler
- Java 5 JDK
- JStatCom
- JMatlab/Link

5 Installation

5.1 Demo

The demo can be started by clicking on `app.bat` and it shows an example usage for the estimation of a GARCH model in a time series analysis context.

The easiest way to create a project with JStatCom and JMatlab/Link is to unzip the file `jmatlab_demo.zip` and to import it as an Eclipse project. This will automatically include the required jar files and the directory `jmatlab`, which holds JMatlab/Link. One may rename the project and packages and use the provided Java source files as a starting point for Matlab calls from within Java.

5.2 JStatCom

Using JStatCom in a Java project is as simple as including `jstatcom.jar` and a few 3rd party libraries to the classpath.

JStatCom is needed for the communication with Matlab dlls because it offers the Matlab Engine that handles the details of each call, like the conversion between Java and Matlab types. JStatCom also offers various other features, but for the basic Matlab functionality, only the Engine system and the Type system are needed.

5.3 JMatlab/Link

JMatlab/Link must reside in the subdirectory `jmatlab` of the working directory of your program. It has the following structure:

- `engine_config.xml` - configuration file for the Matlab engine
- `jmlabmaster.dll` - dll loaded by Java
- `jmlabslave.exe` - Matlab call runner
- `matlabtest.ctf` - ctf file belonging to `matlabtest.dll`
- `matlabtest.dll` - test dll, only for unit tests
- `MATLAB Component Runtime` - directory with the MATLAB Component Runtime, it must be included when programs should be run without a Matlab installation
- `matlabtest_mcr` - generated by Matlab when `matlabtest.dll` was loaded for the first time
- `matlabtest_sources` - Matlab sources for the demo and tests, see `compile.txt` for how they can be compiled

5.3.1 MATLAB Component Runtime (MCR)

The Matlab generated dlls can only be run with the same version of Matlab or the MCR under which they have been generated. JMatlab/Link must be adjusted to use the correct version of the runtime, see Section 5.4

The MATLAB Component Runtime is shipped with the Matlab Compiler and allows developers to make it availabe to users of their Matlab-based software. It is needed for any calls on the Matlab generated dlls when:

- no Matlab installation is available
- the installed Matlab version does not exactly match the Matlab version with which the dlls were generated

JMatlab/Link allows to put the MCR in a subdirectory of the project directory. This way it can be shipped and installed with the software. No separate installation of the MCR is needed, nor do any environment variables have to be set.

To extract the MCR it should be installed in the directory `jmatlab/MATLAB Component Runtime`. The MCR installer can be found under `MATLAB701/toolbox/compiler/deploy/win32/MCRInstaller.exe`.

5.4 Configuring JMatlab/Link

JMatlab/Link must be configured to match the correct Matlab or MCR version. The current default setting is 7.1.

JMatlab/Link can be configured by editing the file `jmatlab/engine_config.xml` with a text editor. The following settings can be adjusted:

- `DLL_NAME` - this must contain the name of the Matlab dll that should be loaded, the default is `mclmcrrt71` for MCR version 7.1
- `RUNTIME_SUBDIR` - if the MCR is installed in a subdirectory of `jmatlab` the correct path to the `bin` directory must be given, it usually contains version information, the default is `MATLAB Component Runtime/v71/runtime/win32`
- `MATLAB_COMPILER_VERSION` - the value should start with `4`, otherwise this option is currently ignored
- `SHARED_MEM_SIZE` - sets the amount of memory that is reserved for the communication between Java Matlab, it should only be increased if very large arrays are used as function or return parameters (more than 1000 x 1000 elements).

6 Calling Matlab from Java

All Matlab generated dlls must reside together with their ctf file in the directory `jmatlab`**.**

Calling Matlab routines requires the following steps:

- load the user-defined Matlab dll
- prepare the input arguments as instances of `com.jstatcom.model.JSCData`
- prepare empty `JSCData` instances to hold return parameters from the Matlab call
- invoke the Matlab call with the correct function name and Java argument/return parameters that match the corresponding Matlab type

```
// generate random data for input arrays
JSCNArray in1 = new JSCNArray("in1", UMatrix.rndu(100, 50));
JSCNArray in2 = new JSCNArray("in2", UMatrix.ones(30, 2));

// prepare empty output arrays
JSCNArray out1 = new JSCNArray("out1");
JSCNArray out2 = new JSCNArray("out2");

// get reference to Matlab engine
final Engine mlab = EngineTypes.MATLAB_COM.getEngine();

// load the user library, this has no effect on subsequent calls
mlab.load("matlabtest", MatlabLoadTypes.USERLIB, null);

// call the Matlab routine "darraytest" with in1, in2
// out1, out2 hold the results
mlab.call("darraytest", new JSCData[] { in1, in2 },
             new JSCData[] { out1, out2 });
```

JStatCom allows to encapsulate engine calls in a class `com.jstatcom.engine.PCall`. This helps to separate calling logic from other code (GoF *Command* pattern). Furthermore it provides a consistent error handling scheme and the ability to run calculations in a separate thread. An example is provided with the class `com.jstatcom.jmatlabdemo.GARCHPCall`.