Writing Matlab Code for CARMEN Services

MATLAB is an extremely popular numerical computing environment maintained by Mathworks Inc. Licensing issues constrain us to the use of matlab compiled executables, which are created via the matlab toolbox. The subsequent executable can be ran from the command line if the correct matlab run-time library is installed. We currently have Matlab 7.6.0 installed on the Carmen servers. The matlab code must be compiled on a machine similar to the architecture of the carmen servers (64-bit AMD Linux).

Analysis tools are typically created as functions in the matlab environment, however to work as a command line we again need a top-level wrapper for these functions, and to handle input and output.

Below is an example of some matlab code written as a command line (see the general rules). This example takes three input parameters and returns a reference to an output file. The three input parameters are; the input filename, the name of the output file, and a numeric value. The algorithm has been left blank for clarity.

```
% top-level function
function myFunction(infile, outfile, value)

% convert input string to numeric
valueNum  = str2num(value);

% load input mat file
load(infile);

% do something useful here
. . .

% save output file
save(outfile);

% disp output filename to screen/stdout
opStr = ['<output>' outfile '</output>'];
disp(opStr);
return
```

Not that the matlab code handles any file access. If an output file is written in the matlab code, the reference must be passed out as the service output. This allows the file reference to be passed to the next service (if necessary).

Should your algorithm return more than one output value, it can be easily accomodated by printing the output strings in a comma-seperated list, i.e.

```
% disp output filename to screen/stdout
opMsg = ['<output>' outputfilename0 ',' outputfilename1 ','
num2str(MaxFrequency) '</output>'];
disp(opMsg);
return
```

Graphical plots can be included in your service. Here is an example of some matlab code for a service that generates a graph, and dumps it to a file. The name of the output file is the only input parameter (not really realistic). We finish with exit this time as it avoids errors when running the compiled executable.

```
function test(outfile)
% create a random vector
a=randn(1,100);
% generate a new figure, and plot something
figure ;
plot(a);
% save the most recent figure to file (whose name was specified as an
input)
print ('-djpeg',outfile);
% display the name of the file
close all force
opStr = ['<output>' outfile '</output>'];
disp(opStr);
end
```

It is important when writing code that makes use of matlab's graphical capabilities, that the top level function has all its figures closed with "close all force". This avoids the following error/warning when running the compiled executable on a machine with no DISPLAY setup (such as the Carmen service nodes). So if you test your graphical matlab service and see the following output, then add "close all force"!

```
If your application has terminated unexpectedly, please note that
applications
generated by the MATLAB Compiler terminate when there are no visible figure
windows.
See the documentation for WaitForFiguresToDie and WAITFORCALLBACKS for more
information.

Warning: Class
    'graph2d.lineseries'
in use at MCR termination.

If your application has terminated unexpectedly, please note that
applications generated
by the MATLAB Compiler terminate when there are no visible figure windows.
See the
documentation for WaitForFiguresToDie and WAITFORCALLBACKS for more
information.
```

You may notice that when testing this as a compiled executable that the application is interactive. This breaks the basic rules for wrapping. However, when run with no valid X-Server session (as it will be when deployed as a web service) it becomes non-interactive.

If you see the following warning when running your compiled matlab application from the command line, when NO DISPLAY environment is set, it's ok, just ignore it as the service will handle it automatically.

```
Warning: No display specified.  You will not be able to display graphics on
the screen.
```

Creating Matlab Compiled Executables

Start the Matlab Environment.

For first time use, you need to setup the Matlab compiler, so type the following in the Matlab command window

mbuild -setup

Then from Matlab's start menu, select:

START > MATLAB > MATLAB Compiler > Deployment Tool (deploytool)

1. In the deployment tool's toolbar, select a "new deployment project".
2. You will then get the choice of creating a "Standalone Application", a "C Shared Library", or a "C++ Shared Library". Select "Standalone Application".
3. Give the project a name – this will be create a folder of the same name and on successful compilation, will also be the name of the resultant compiled executable.

You will now see a directory tree of:

- Main function
- Other files
- C/C++ files

Your main top-level m-file code needs to go into the main function folder.

Additional files containing support functions can be dragged into the "Other files" section.

If you are using a mex C-function – first compile the c code either on the unix or matlab command line terminal. The resultant mexa64 file can be deployed in the **other files** folder.

The project can now be built using the toolbar icon.

Looking inside your project folder, inside the "distrib" subfolder, you should find a file with the same name as your project (no file extension). This is the standalone Matlab compiled executable. This file can be run from the command line, assuming the runtime library is available.