MATLAB code compilation

One of the best ways to earn some substantial performance increase of MATLAB code is to compile it using the MATLAB Complier. This is especially beneficial when the M-files contain some non-interpreter-friendly-statements such as “for” or “while” loops. MATLAB Compiler is a part of the MATLAB product. It takes M-files as input and generates C or C++ source code as output. One can also use MATLAB Compiler to generate C source code for building MEX-files. MEX (MATLAB Executable) files are dynamically linked subroutines produced from C source code that, when compiled, can be run from within MATLAB in the same way as M-files or built-in functions. MEX files interface pre-existing C code with MATLAB without rewriting the code in MATLAB.

MATLAB Compiler supports almost all the capabilities of MATLAB. However, there are some limitations that you should be aware of. For example, MATLAB Compiler cannot compile:

- Script M-files (non functions M-files).
- M-files that dynamically name variables to be loaded or saved.

For a complete list of limitations, please consult the MATLAB Compiler documentation.

To compile your code, please follow the steps below:

1.
Verify that MATLAB Compiler is installed on your system (it should be) by typing *mcc* on the MATLAB prompt. If it’s not installed, install it.

2.
Verify that you have an ANSI C or C++ compiler installed on your system (for example, Microsoft Visual Studio 6). Note that MATLAB includes an ANSI C compiler (*Lcc*) that is suitable for use with the MATLAB Compiler.

3.
MATLAB Compiler uses the *mex* utility to generate MEX-files from C source code. On systems where there is exactly one C or C++ compiler available to you, the *mex* utility automatically configures itself for the appropriate compiler. So, for many users, to create a MEX-file, you can simply enter:

mex filename.c

On systems where there is more than one C or C++ compiler, the *mex* utility lets you select which of the compilers you want to use. Once you choose your C or C++ compiler, this compiler becomes your default compiler and you no longer have to select one when you compile MEX-files. For example, if your system has both the Lcc and Microsoft Visual C/C++ compilers, when you enter for the first time:

mex filename.c

you are asked to select which compiler to use:

mex has detected the following compilers on your machine:

[1] : Lcc C version 2.4 in C:\MATLABR12\sys\lcc
[2] : Microsoft Visual C/C++ version 6.0 in C:\Program Files\Microsoft Visual Studio\VC98\Bin

[0] : None

Please select a compiler. This compiler will become the default:

Select the desired compiler by entering its number. You are then asked to verify the information.

4.
It's now time to verify that your system can create MEX-files from C source code. The *<matlab directory>\extern\examples\mex* directory contains C source code for the example *yprime.c*. To verify that your system can create MEX-files, enter at the MATLAB prompt

```
cd([matlabroot '\extern\examples\mex'])
mex yprime.c
```

This should create the yprime.dll MEX-file. MEX-files created on Windows 95/98/2000 or NT always have the extension DLL. You can now call yprime as if it were an M-function. For example:

```
yprime(1,1:4)

=> ans =
    2.0000   8.9685   4.0000   -1.0947
```

Note that included in MATLAB is an add-in for Visual Studio that lets you work easily within the Microsoft Visual C/C++ environment to create and debug MEX-files.

5.
The last thing to verify is that MATLAB Compiler is correctly installed. For example, type the following at the MATLAB prompt:

```
mcc -x invhilb
```

After a short delay, this command should complete and display the MATLAB prompt. Next, at the MATLAB prompt, type:

which invhilb

The *which* command should indicate that *invhilb* is now a MEX-file; the mex command should have created the file *invhilb.dll*. Finally, use *invhilb.dll*; for example, type at the MATLAB prompt:

invhilb(10)

6.
If anything went wrong during the above process, please contact the lab application engineer or one of the lab assistants. If everything went correctly, you are now ready to compile your M-files.

Good Luck!