MATLAB Tutorial

Chapter 8. MATLAB compiler

The previous chapters have discussed programming within the MATLAB environment. It has been noted that MATLAB is an interpreted language, meaning that each command is converted to machine-level instructions one-by-one during execution. While this allows one to program in both interactive and batch mode, the extra overhead required to convert the commands at run-time is not desired. Also, any programs written in MATLAB can only be run on computers that have a copy of MATLAB, so portability is limited. MATLAB includes an optional compiler to circumvent these problems by converting m-files to C or C++ code, and optionally linking this code with its mathematics and graphics libraries to produce a stand-alone executable that may be run, without the interpretation overhead, on any machine with a compatible operating system platform. In this section, we demonstrate the MATLAB compiler to produce a stand-alone executable from the simple example of section 6.4. Note that the program containing the main program has been rewritten from the previous script file version since the MATLAB compiler only works with function m-files. The first file, a script file called make_file.m, is executed from the interactive prompt to perform the compilation; alternatively, the command mcc ... can be entered manually.

make_file.m

This MATLAB script m-file calls the compiler to convert the MATLAB source code files for make_plot_trig to C, link the object files with the MATLAB graphics library, and then produce a stand-alone executable.

Kenneth Beers
Massachusetts Institute of Technology
Department of Chemical Engineering

7/31/2001

mcc -B sgl ...
make_plot_trig ...
plot_trig_1 ...
trig_func_1 ...
get_input_scalar ...
assert_scalar

make_plot_trig.m (main program file)

make_plot_trig.m

This MATLAB m-file makes a plot of the general function
f(x) = a*sin(x) + b*cos(x)
for user-selected values of a and b.

Kenneth Beers
Massachusetts Institute of Technology
Department of Chemical Engineering

7/31/2001

function iflag_main = make_plot_trig();

iflag_main = 0; signifies no completion

```
disp('RUNNING make_plot_trig ...');
disp(' ');
disp('This program produces a plot in [0,2*pi]');
disp('of the function : ');
disp('f(x) = a*sin(x) + b*cos(x)');
disp('for user-input values of the real scalars a and b');
disp(' ');
```

The following code asks the user to input values of a and b, and then uses plot_trig to plot trig_func_1 by including the function name as an argument in the list.

```
prompt = 'Input a : ';
check_real=1; check_sign=0; check_int=0;
a = get_input_scalar(prompt, ...
check_real,check_sign,check_int);

prompt = 'Input b : ';
check_real=1; check_sign=0; check_int=0;
b = get_input_scalar(prompt, ...
check_real,check_sign,check_int);
```

We now call the routine that produces the plot.

```
func_name = 'trig_func_1';
plot_trig_1(func_name,a,b);
```

We now require the user to strike a key before exiting the program.

```
pause

iflag_main = 1;

return;
```

plot_trig_1.m

```
function iflag = plot_trig_1(func_name,a,b);
```

iflag = 0; signifies no completion

First, create an x vector from 0 to 2*pi

```
num_pts = 100;
x = linspace(0,2*pi,num_pts);
```

Next, make a vector of the function values. We evaluate the argument function indirectly using the "feval" command.

```
f = linspace(0,0,num_pts);
for i=1:num_pts
f(i) = feval(func_name,x(i),a,b);
end
```

Then, we make the plot.

```
figure;
plot(x,f);
xlabel('Angle (radians)');
ylabel('Function value');
```

```
return;
```

trig_func_1.m

```
function f_val = trig_func_1(x,a,b);
f_val = a*sin(x) + b*cos(x);

return;
```