Matlab coder

Curt Da Silva

SLIM

University of British Columbia

Matlab

Great for prototyping algorithms

- code matches math really nicely

Implementation of production-ready codes

- ehhh

Matlab

What are the main impediments to efficiency in Matlab?

- for loops have overhead, nested for-loops especially so
- checking bounds for array access
- vectorization - efficient, but memory intensive, produces hard to read code

Matlab Coder

Allows you to create C source file from Matlab m files

- no references to Matlab runtime - functions can be integrated in to a static/dynamic library that can be integrated in to other codes

Automatic mex file generation

- just a wrapper to call the generated C function

Matlab Coder

Automatic use of OpenMP parallelization with parfor loops (for supported compilers)*

Get speedups from plain matlab code basically for free

- I would recommend using this for core computational kernel code, not so much setup code

Only supports a subset of the MATLAB language

- few toolboxes, not all commands supported, etc.

Matlab Coder

Only supports a subset of the MATLAB language

- things like ‘fft’, ‘svd’ require the MATLAB runtime
- does support automatic code substitution, so you can call other implementations of the MATLAB-requiring functions

Matlab Coder

Annoying things

- we only have licenses on our Macs, not on the cluster

- Matlab on OSX doesn't support GCC / I haven't found a way to make it use the latest CLANG compiler
  - == no automatic openMP support for the compiled functions on mac =(
  - you can add this in manually, but it's annoying

- the GUI interface is atrocious/buggy, so just use the attached script

Matlab Coder

Annoying things

- because Matlab is so flexible with data types and C is not at all, you'll basically have to declare everything as floating point, so the coder won't complain
  - potential slowdowns from double/single -> integer conversions in the C code, which have to be rectified

Structure of codegen/

lib/

- build files for creating a static library

mex/

- build files for creating a static library + mex wrapper

Structure of codegen/mex

<<function_name>>_mex.sh

- shell script that runs the compilation process (can change architecture, matlab location,etc.)

<<function_name>>_mex.mk

- gmake file specifying starting directory (modifiable), Matlab location (modifiable), other compilation parameters (don't touch these)

<<function_name>>_mex.mki

- environment variables generated from setEnv.sh , you can modify them but it won't do anything because they'll just be overwritten again

Structure of codegen/mex

setEnv.sh

- environment variables for the build are set here
- if you want to use different compilers, different compilation/linking flags, etc., set those all up here

<<function_name>>_mex.c

- generated C source file from Matlab Coder, main computation happens here
- jumping off point for your own optimizations

Matlab coder example

```
% Matlab function to compile
srcfile = 'Helm3dmvp.m';

% various coder options, see the documentation for details
cfg = coder.config;
cfg.MATLABSourceComments = true;
cfg.EnableDebugging = false;
cfg.IntegrityChecks = false; %I got a 2x speedup turning
these off
cfg.ResponsivenessChecks = false;
cfg.EnableOpenMP = true; % won't do anything with old clang
```

Matlab coder example

```
% setup type + size of arguments
% here wn is a n x 1 complex vector, where n can be anything
wn = coder.typeof(complex(double(0),double(0)),[Inf 1]);

% h, n are 3 x 1 double vector
h = coder.typeof(double(0),[3 1]);
n = coder.typeof(double(0),[3 1]);
% npml is a 2 x 3 double matrix
npml = coder.typeof(double(0),[2 3]);
```

```
% x is a n x 1 complex vector, n can be anything
x = coder.typeof(complex(double(0),double(0)),[Inf 1]);

% mode is a double scalar
mode = coder.typeof(double(0),1,1);

% all of the arguments to this function
args = {wn,h,n,npml,x,mode};

% run Matlab Coder
codegen(srcfile,'-config',cfg,'-args',args);
```

Matlab coder example

```
n = [70;70;70]; npml = 10*ones(2,3); h = [25;25;25];
wn = complex(rand(prod(n),1)+0.5,rand(prod(n),1)+0.5);
q = complex(randn(prod(n),1),randn(prod(n),1));

tic,y = Helm3dmvp(wn,h,n,npml,q,1);disp(toc);
tic,y1 = Helm3dmvp_mex(wn,h,n,npml,q,1);disp(toc);
disp(norm(vec(y)-vec(y1)));
```

Output:

11.5248 m-file, non vectorized

0.1533 Matlab coder C non vectorized

0

Matlab coder example

```
tic,y2 = Helm3dmvp_v(wn,h,n,npml,q,1);disp(toc); (vectorized code)
```

Output:

0.7295 vectorized m-file

Matlab coder example

```
tic,y2 = Helm3dmvp_v(wn,h,n,npml,q,1);disp(toc); (vectorized code)
tic,y3 = Helm3dmvp_v_mex(wn,h,n,npml,q,1);disp(toc); (Matlab Coder)
```

Output:

0.7295	vectorized m-file
0.6338	vectorized m-file -> C

Matlab coder example

```
tic,y2 = Helm3dmvp_v(wn,h,n,npml,q,1);disp(toc); (vectorized code)
tic,y3 = Helm3dmvp_v_mex(wn,h,n,npml,q,1);disp(toc); (Matlab Coder)
tic,yref = Helm3dmvp_forw_mex(wn,h,n,npml,q,1);disp(toc); (hand coded C)
```

Output:

0.7295	vectorized m-file
0.6338	vectorized m-file -> C
0.1120	hand coded C

Summary

MVP time (s)

11.5248	m-file - non vectorized
0.1533	Matlab coder C - non vectorized
0.7295	m-file - vectorized
0.6338	Matlab coder C - vectorized
0.1120	hand coded C

Matlab Coder

Low effort way to speed up (non-vectorized) Matlab code

Good jumping off point for adding in your own optimizations

Wayyyy easier than coding things in C from scratch

Generated code is mostly readable, with matlab code as comments included

Matlab Coder

This presentation will be on my internal blog at

https://slimgroup.eos.ubc.ca/users/curtd/weblog/b8c6c/Matlab_Coder.html

if you want to copy and paste code/play around with scripts