Introduction to Programming in Matlab with MEX

Math 663

Alexey Zimin

MATLAB

- MATLAB (by Mathworks) is a good development platform for matrix analysis algorithms.
- It is heavily optimized for vector operations.
  - Good for fast calculations on vectors and matrices.
  - Bad if you can not state your problem as a vector problem.
    - Slow implementations of sequential programs
    - For-loops are slow.

What are MEX-files?

- MEX stands for MATLAB Executable.
- MEX-files are a way to call your custom C or FORTRAN routines directly from MATLAB as if they were MATLAB built-in functions.
- Mex-files can be called exactly like M-functions in MATLAB.
- Here, all code examples will be presented in C.

Reasons for MEX-files

- The ability to call large existing C or FORTRAN routines directly from MATLAB without having to rewrite them as M-files.
- Speed; you can rewrite bottleneck computations (like for-loops) as a MEX-file for efficiency.
- Parallelism – you can write multi-threaded C code for operations that cannot be simply vectorized

The mxArray

- All Matlab variables are stored as Matlab arrays. In C, the Matlab array is declared to be of type **mxArray**, which is defined by a structure.
- The structure contains:
  - Its type.
  - Its dimensions.
  - The data associated with the array.
  - If numeric, whether real or complex.
  - If sparse, its nonzero indices.
  - If a structure or object, more info

MEX data types

- Fundamental types: double, char, logical, int, cell, struct
- Derived Types (represented in C by the **mxArray** structure):
  - Numeric
    - Complex double-precision nonsparse matrix.
      - Complex.
      - Real (pointer to vector of imaginary elements points to NULL).
    - Single-precision floating point, 8-,16-, and 32-bit integers, both signed and unsigned, real and complex.
  - Strings (strings are not null terminated as in C).
  - Sparse Matrices, Cell Arrays, Structures, Objects, Multidimensional Arrays.

Indexing of mxArrays

- Indexing : Column wise, as in MATLAB

  0 3 6

  1 4 7

  2 5 8

MX Functions

- The collection of functions used to manipulate mxArrays are called MX-functions and their names begin with mx.
- Examples:
  - mxArray creation functions:

    mxCreateNumericArray, mxCreateDoubleMatrix, mxCreateString, mxCreateDoubleScalar.
  - Access data members of mxArrays:

    mxGetPr, mxGetPi, mxGetM, mxGetN.
  - Modify data members:

    mxSetPr, mxSetPi.
  - Manage mxArray memory:

    mxMalloc, mxCalloc, mxFree, mxDestroyArray

MEX Functions

- The collection of functions used to perform operations back in Matlab are called MEX-functions and begin with mex.
- Examples:
  - mexFunction: Gateway to C.
  - mexEvalString: Execute Matlab command.
  - mexCallMatlab: Call Matlab function(.m or .dll) or script.
  - mexPrintf: Print to the Matlab editor.
  - mexErrMsgTxt: Issue error message and exit returning control to Matlab.
  - mexWarnMsgTxt: Issue warning message

Components of a MEX-file

- A gateway routine, mexFunction, that interfaces C and MATLAB data
- A computational routine, called from the gateway routine, that performs the computations that the MEX-file should implement
- Preprocessor macros, for building platform independent code

The mexFunction: Gateway to Matlab

- The main() function is replaced with mexFunction.

```
#include "mex.h"
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{//code that handles interface and calls
//to computational function
return; }
```

- mexFunction arguments:
  - nlhs: The number of lhs (output) arguments.
  - plhs: Pointer to an array which will hold the output data, each element is type **mxArray**.
  - nrhs: The number of rhs (input) arguments.
  - prhs: Pointer to an array which holds the input data, each element is type const **mxArray**

Some important points

- The parameters **prhs**, **plhs**, **nrhs** and **nlhs** are required.
- The header file, **mex.h**, that declares the entry point and interface routines is also required.
- The name of the file with the gateway routine will be the command name in MATLAB.
- The file extension of the MEX-file is platform-dependent.
- The **mexext** function returns the extension for the current machine.
- An extra important point: MATLAB is 1-based and C is 0-based !!!

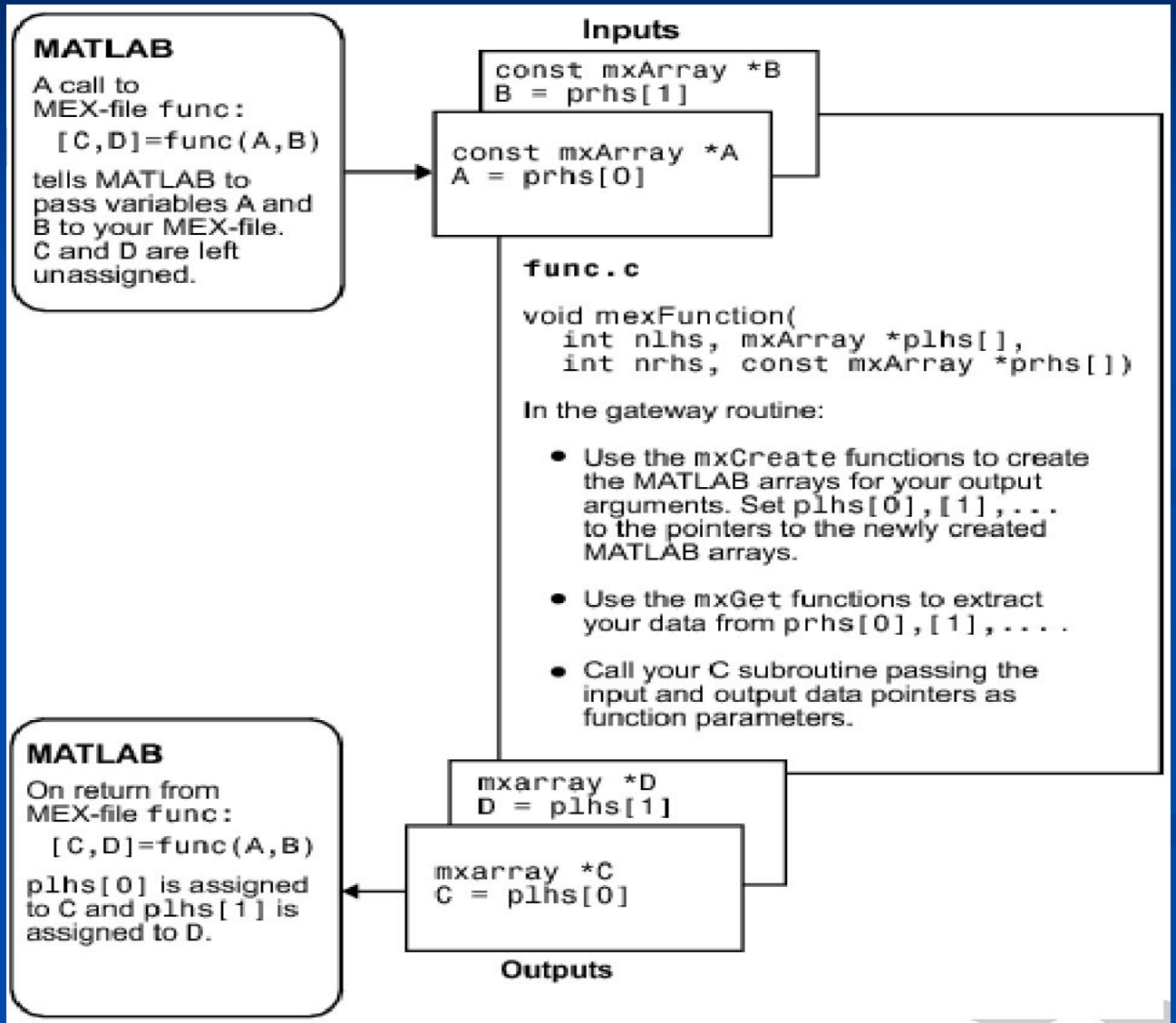
The computational routine

- The computational routine is called from the gateway routine
- It is a good idea to place the computational routine in a separate subroutine although it can be included in the gateway routine

Input and output, I/O

- **[C,D] = myfunc(A,B)**
- Get pointers to A and B
  - **mxGetPr(prhs[0])**
  - **mxGetPr(prhs[1])**
- Allocate memory for C and D
  - **mxCreate* (NumericArray, DoubleMatrix, ...)**
- Get pointers to C and D
  - **mxGetPr(plhs[0])**
  - **mxGetPr(plhs[1])**

Overview of the communication between MEX and MATLAB



Compiling MEX-files

- Compile your mex function on the MATLAB command line using the mex command:

mex myfunc.c

- Easy compilation of required files by adding them on the command line

mex myfunc.c special.cpp

- Compile outside MATLAB in your favourite development environment

Output files

- .dll
- .mexa64
- .mexw32
- .mexglx

Calling MEX-functions from Matlab

- You can call MEX-files exactly as you would call any M-function.
- If you call a MATLAB function the current working directory and then the MATLAB path is checked for the M-or MEX-function.
- .MEX-files take precedence over M-files when like-named files exist in the same directory.
- Help text documentation is read from the .m file with same name as the MEX-file. Add your usage tips in the .m file.

Example 0

```
#include "/software/Linux64/matlab-7.5/extern/include/mex.h"

/* the gateway function */
void mexFunction(int nlhs,
                 mxArray *plhs[],
                 int nrhs,
                 const mxArray *prhs[])
{
mexPrintf("Hello World\n");
}
```

Example 1

```
#include "/software/Linux64/matlab-7.5/extern/include/mex.h"
#include <math.h>

double mod(double,double);/* matlab-like mod function */

void mexFunction(int nlhs,
                 mxArray *plhs[],
                 int nrhs,
                 const mxArray *prhs[])
{
double *y,sum=0;
mxArray *sv;
int i,m,n;

/*get the size of  the input matrix*/
m=mxGetM(prhs[0]);
n=mxGetN(prhs[0]);
mexPrintf("Input matrix is %d by %d\n",m,n);
```

Example 1 (cont)

```
/* one output matrix */
plhs[0]=mxCreateDoubleMatrix(1,1,mxREAL);

/*initialize*/
sv=mxCreateDoubleMatrix(m,1,mxREAL);

/* y is the pointer to the array of output matrix */

mexCallMATLAB(1,&sv,1,&prhs[0],"svd");
y=mxGetPr(sv);
for(i=0;i<m;i++)
        sum = sum+mod(y[i],1);

y=mxGetPr(plhs[0]);
*y=sum;
mxDestroyArray(sv);
return;
}
```

Example 1 (cont)

```
double mod(double x,double y)
{

  if(y!=0.)
    return(x - y*floor(x/y));
  else
    return(x);
}
```

Documentation

- On www.mathworks.com > Support > Product documentation > MATLAB
  - External interfaces: Creating C Language MEX-Files