Centre for Image Analysis
Swedish University of Agricultural Sciences
Uppsala University

An introduction to MATLAB MEX-files

Maria Axelsson
maria@cb.uu.se

MATLAB

- MATLAB (by Mathworks) is a good development platform for image 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! Especially, 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.

Components of a MEX-file

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

The gateway routine

- The name of the gateway routine must be **`mexFunction`**.

`prhs`	An array of right-hand input arguments.
`plhs`	An array of left-hand output arguments.
`nrhs`	The number of right-hand arguments, or the size of the prhs array.
`nlhs`	The number of left-hand arguments, or the size of the plhs array.

```
void mexFunction(
    int nlhs, mxArray *plhs[],
    int nrhs, const mxArray *prhs[])
{
    /* more C code ... */
}
```

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

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

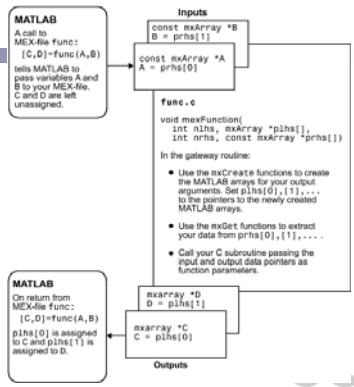
- MATLAB is 1-based and C is 0-based.

Input and output, I/O

- [C,D] = func(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])`**

I/O

- Overview of the communication between MEX and MATLAB



I/O example

```
unsigned short *pind_l,*pind_r;
int number_of_dims, nelem;
const int  *dim_array;
...

number_of_dims = mxGetNumberOfDimensions(prhs[0]);
dim_array = mxGetDimensions(prhs[0]);
nelem = mxGetNumberOfElements(prhs[0]);

plhs[0] = mxCreateNumericArray(number_of_dims,dim_array,
mxUINT16_CLASS, mxREAL);

pind_l = (unsigned short *) mxGetPr(plhs[0]);
pind_r = (unsigned short *) mxGetPr(prhs[0]);

/* call computational routine */
myfunction(pind_l,pind_r, ... );
```

mxArray

- All scalars, vectors, matrices etc in MATLAB are represented in mxArrays

- mxCreateCellArray
- mxCreateCellMatrix
- mxCreateCharArray
- mxCreateCharMatrixFromStrings
- mxCreateDoubleMatrix
- mxCreateDoubleScalar
- mxCreateLogicalArray
- mxCreateLogicalMatrix
- mxCreateLogicalScalar
- mxCreateNumericArray
- mxCreateNumericMatrix
- mxCreateSparse
- mxCreateSparseLogicalMatrix
- mxCreateString
- mxCreateStructArray
- mxCreateStructMatrix

mxArray suitable for images

- 2D
  - **mxCreateNumericArray**
  - **mxCreateDoubleMatrix**
- 3D
  - **mxCreateNumericArray**
- Indexing
  - Column wise, as in MATLAB

0	3	6
1	4	7
2	5	8

mx and mex

- Routines in the API that are prefixed with mx allow you to create, access, manipulate, and destroy mxArrays.
- Routines prefixed with mex perform operations back in the MATLAB environment.
  - Note: mex routines are only available in MEX-functions

Checking of input types

- Input arguments needs to be checked
  - **mxIsDouble, mxIsUint16, …**

```
/* Check data type of input argument. */
  if (!(mxIsUint16(prhs[0]))) {
    mexErrMsgTxt("Input array must be of type
   uint16.");
  }
```


2D

```
#include "mex.h"

void xtimesy(double x, double *y, double *z, int m, int n)
{ ... }

/* the gateway function */
void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[])
{
  double *y,*z;
  double  x;
  int     status,mrows,ncols;

  /*  check for proper number of arguments */
  if(nrhs!=2)
    mexErrMsgTxt("Two inputs required.");
  if(nlhs!=1)
    mexErrMsgTxt("One output required.");

  /* check to make sure the first input argument is a scalar */
  if( !mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) ||
      mxGetN(prhs[0])*mxGetM(prhs[0])!=1 ) {
    mexErrMsgTxt("Input x must be a scalar.");
  }
```

2D

```
  ...

  /*  get the scalar input x */
  x = mxGetScalar(prhs[0]);

  /*  create a pointer to the input matrix y */
  y = mxGetPr(prhs[1]);

  /*  get the dimensions of the matrix input y */
  mrows = mxGetM(prhs[1]);
  ncols = mxGetN(prhs[1]);

  /*  set the output pointer to the output matrix */
  plhs[0] = mxCreateDoubleMatrix(mrows,ncols, mxREAL);

  /*  create a C pointer to a copy of the output matrix */
  z = mxGetPr(plhs[0]);

  /*  call the C subroutine */
  xtimesy(x,y,z,mrows,ncols);

}
```

3D

```
void mexFunction(int nlhs, mxArray *plhs[],
                 int nrhs, const mxArray *prhs[])
{
  unsigned int *out_reg, *in_reg;
  int number_of_dims, nelem, zMax;
  const int  *dim_array;

  if (!(mxIsUint32(prhs[0]))) {
    mexErrMsgTxt("First input array must be of type uint32.");
  }

  number_of_dims = mxGetNumberOfDimensions(prhs[0]);
  dim_array = mxGetDimensions(prhs[0]);

  zMax = dim_array[2];

  plhs[0] = mxCreateNumericArray(number_of_dims,dim_array,
                 mxUINT32_CLASS, mxREAL);
  out_reg = (unsigned int *) mxGetPr(plhs[0]);

  in_reg = (unsigned int *) mxGetPr(prhs[0]);
  nelem = mxGetNumberOfElements(prhs[0]);

  remove_function(out_reg,in_reg,nelem,dim_array[1],dim_array[0],zMax);
}
```

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

Setting up the environment

- **mex -setup**

  Choose compiler (see example)

  Default is a C compiler. Add others for C++.
- Compatible compilers http://www.mathworks.com/support/tech-notes/1600/1601.html

mex -setup

```
>> mex -setup
Please choose your compiler for building external interface (MEX)
   files:

Would you like mex to locate installed compilers [y]/n?

Select a compiler:
[1] Lcc C version 2.4.1 in C:\PROGRAM FILES\MATLAB\R2006A\sys\lcc
[2] Microsoft Visual C/C++ version 8.0 in C:\Program
   Files\Microsoft Visual Studio 8
[3] Microsoft Visual C/C++ version 7.1 in C:\Program
   Files\Microsoft Visual Studio .NET 2003

[0] None

Compiler:
```


Calling MEX-functions

- 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.

Documentation

On mathworks.com > Support > Product documentation > MATLAB

- External interfaces
  - Creating C Language MEX-Files
- C and Fortran API Reference

More on MEX

- Call MATLAB or other MEX-functions from MEX
  - `mexCallMATLAB`
  - `mexEvalString`

Pros and cons

- ☑ Fast calculations
- ☑ Easy to learn and use

- ☒ Slow implementation compared to M-files
- ☒ Platform dependent implementation (recompile MEX-files for every platform)

Thank you!

- Thanks for listening to this introduction to MATLAB MEX-files
- Questions?