ETS group meeting

intro to faster matlab code

by

Rob Young

overview

- motivation
- philosophy
- efficient Matlab techniques (tip of iceberg)
- GPU enabled Matlab functions
- parallel for loops
- MEX
- CUDA

motivation

- You don't want to wait for results
- Your labmates don't want to wait for your results

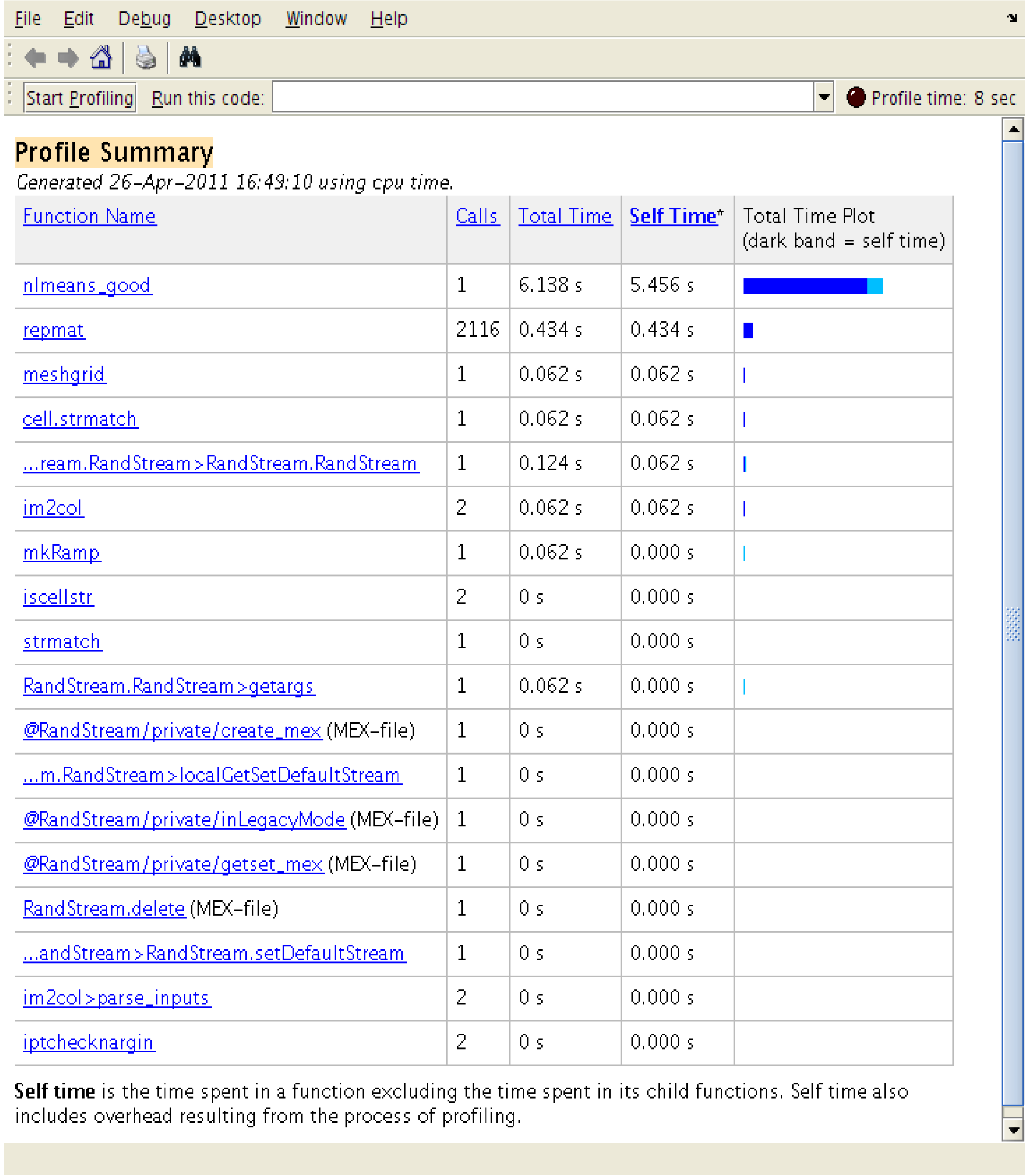
philosophy

“Premature optimization is the root of all evil (or at least most of it) in programming.” --Knuth

- readability is key
  - less errors
  - reusable
- only optimize bottlenecks
  - keep readable code commented

efficient Matlab - profiler

- find bottlenecks:
  1) > profile on
  2) run your code
  3) > profile viewer



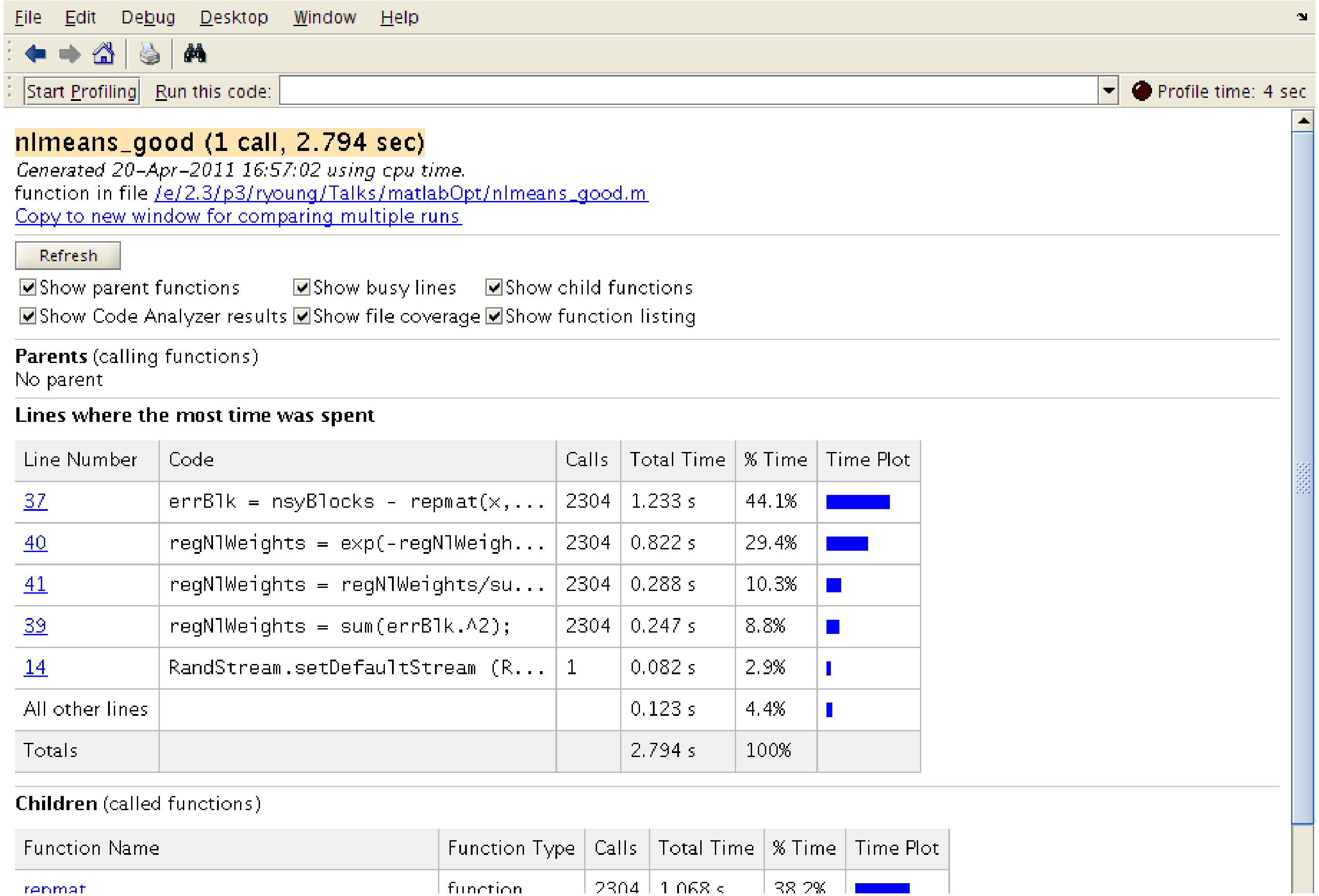
Profile Summary

Generated 26-Apr-2011 16:49:10 using cpu time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
nlmeans_good	1	6.138 s	5.456 s	
repmat	2116	0.434 s	0.434 s	
meshgrid	1	0.062 s	0.062 s	
cell.strmatch	1	0.062 s	0.062 s	
...ream.RandStream>RandStream.RandStream	1	0.124 s	0.062 s	
im2col	2	0.062 s	0.062 s	
mkRamp	1	0.062 s	0.000 s	
iscellstr	2	0 s	0.000 s	
strmatch	1	0 s	0.000 s	
RandStream.RandStream>getargs	1	0.062 s	0.000 s	
@RandStream/private/create_mex (MEX-file)	1	0 s	0.000 s	
...m.RandStream>localGetSetDefaultStream	1	0 s	0.000 s	
@RandStream/private/inLegacyMode (MEX-file)	1	0 s	0.000 s	
@RandStream/private/getset_mex (MEX-file)	1	0 s	0.000 s	
RandStream.delete (MEX-file)	1	0 s	0.000 s	
...andStream>RandStream.setDefaultStream	1	0 s	0.000 s	
im2col>parse_inputs	2	0 s	0.000 s	
iptchecknargin	2	0 s	0.000 s	

Self time is the time spent in a function excluding the time spent in its child functions. Self time also includes overhead resulting from the process of profiling.

Profiler – time spent per line



Start Profiling Run this code: ● Profile time: 4 sec

nlmeans_good (1 call, 2.794 sec)

Generated 20-Apr-2011 16:57:02 using cpu time.

function in file /e/2.3/p3/ryoung/Talks/matlabOpt/nlmeans_good.m

Copy to new window for comparing multiple runs

Refresh

☑ Show parent functions ☑ Show busy lines ☑ Show child functions

☑ Show Code Analyzer results ☑ Show file coverage ☑ Show function listing

Parents (calling functions)

No parent

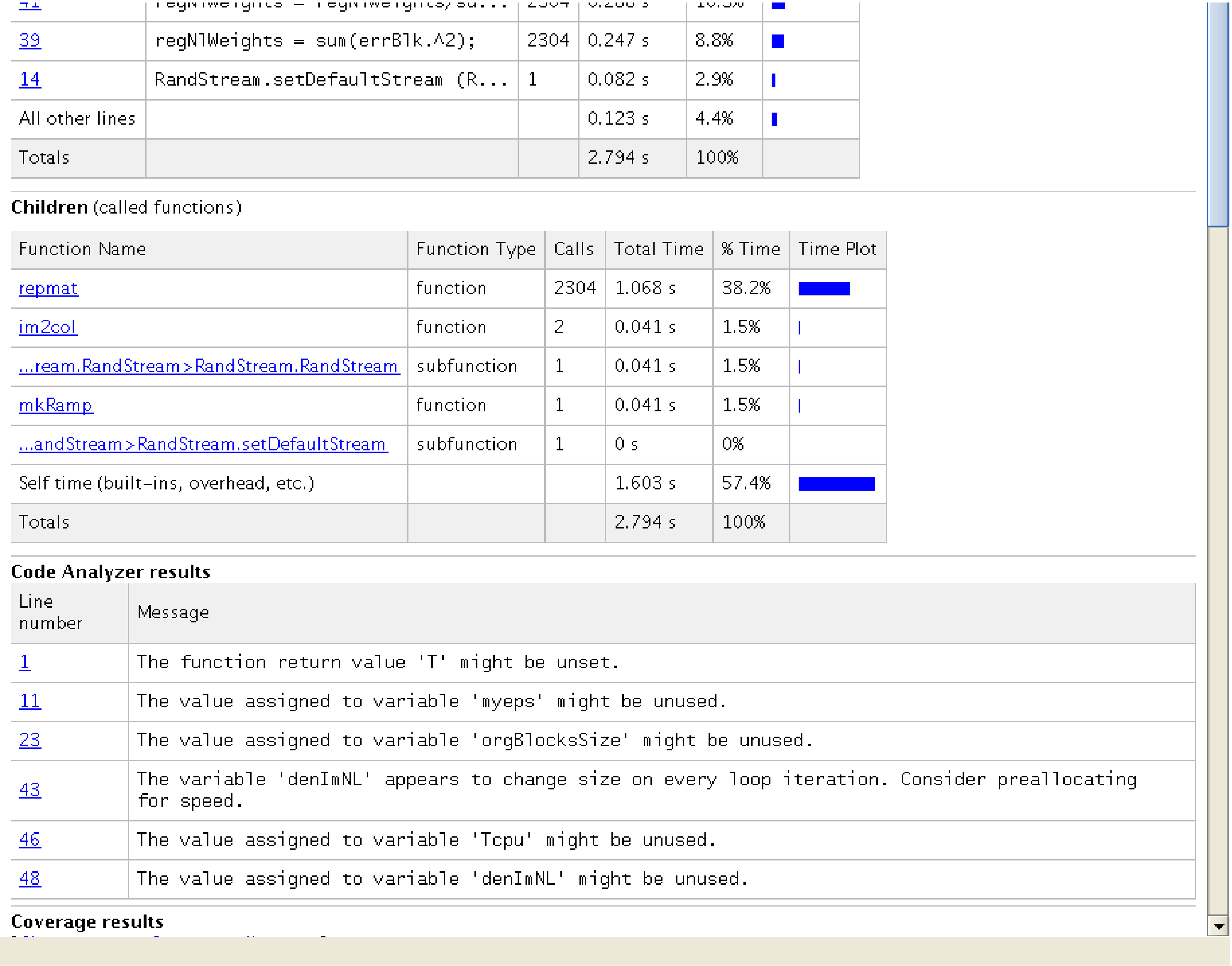
Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
37	`errBlk = nsyBlocks - repmat(x,...`	2304	1.233 s	44.1%	▬
40	`regNlWeights = exp(-regNlWeigh...`	2304	0.822 s	29.4%	▬
41	`regNlWeights = regNlWeights/su...`	2304	0.288 s	10.3%	▪
39	`regNlWeights = sum(errBlk.^2);`	2304	0.247 s	8.8%	▪
14	`RandStream.setDefaultStream (R...`	1	0.082 s	2.9%	▮
All other lines			0.123 s	4.4%	▮
Totals			2.794 s	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Time Plot
repmat	function	2304	1.068 s	38.2%	▬

Profiler – mlint (Code Analyzer)



41	regNlWeights = regNlWeights/su...	2304	0.288 s	10.3%	■
39	regNlWeights = sum(errBlk.^2);	2304	0.247 s	8.8%	■
14	RandStream.setDefaultStream (R...	1	0.082 s	2.9%	▮
All other lines			0.123 s	4.4%	▮
Totals			2.794 s	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Time Plot
repmat	function	2304	1.068 s	38.2%	▬
im2col	function	2	0.041 s	1.5%	|
...ream.RandStream>RandStream.RandStream	subfunction	1	0.041 s	1.5%	|
mkRamp	function	1	0.041 s	1.5%	|
...andStream>RandStream.setDefaultStream	subfunction	1	0 s	0%	
Self time (built-ins, overhead, etc.)			1.603 s	57.4%	▬
Totals			2.794 s	100%	

Code Analyzer results

Line number	Message
1	The function return value 'T' might be unset.
11	The value assigned to variable 'myeps' might be unused.
23	The value assigned to variable 'orgBlocksSize' might be unused.
43	The variable 'denImNL' appears to change size on every loop iteration. Consider preallocating for speed.
46	The value assigned to variable 'Tcpu' might be unused.
48	The value assigned to variable 'denImNL' might be unused.

Coverage results

efficient Matlab - vectorize

For loops are slow in Matlab, so replace with colon (:) or repmat:

```
i = 0;
for t = 0:0.001:1
    i = i + 1;
    y(i) = sin(t);
end
```

with:

```
t = 0:0.001:1;
y = sin(t);
```

efficient Matlab – pre-allocation

- If you are stuck with a for loop then make sure you preallocate:

```
foo = zeros(1,N);
for i = 1:N
    foo(i) = baz(i);
end
```

- otherwise you're reallocating a new array at each iteration

efficient Matlab - In-place operations

- Many Matlab functions support in-place operation on data:

  x = myfunc(x)

  - No memory overhead and no time overhead for allocation.

efficient Matlab – single precision

- Do you really need double precision?
- If not allocate as single precision:
  foo = single(rand(N));
- quick way to cut execution time in half.

  (almost anyway)
- cuts internal representation of variables in half

parallel threads of execution

- Matlab >= 7.4 supports CPU multithreading
  - CPU usage > 100% == CPU multithreading
- Matlab >= 7.11 supports GPU multithreading
- example: independent iterations of for loop
  - pass each job to its own processing core (CPU or GPU)
  - Multiple iterations done at each time step

efficient Matlab – GPU functions

- latest versions of Matlab have limited GPU support:
  - arrayfun, conv, dot, filter, fft, ifft, ldivide, lu, mldivide, …
- data transfer to and from card is slow
- works best with vectorized code

GPU functions - example

```
% move data to GPU
X_gpu = gpuArray(im_cpu);
Y_gpu = gpuArray(filt_cpu);
< perform operations on the GPU >
Z_gpu = ifft( fft(X_gpu) .* fft(Y_gpu) );
Z_cpu = gather(Z_gpu);% pull data off the GPU
```

faster for loops - parfor

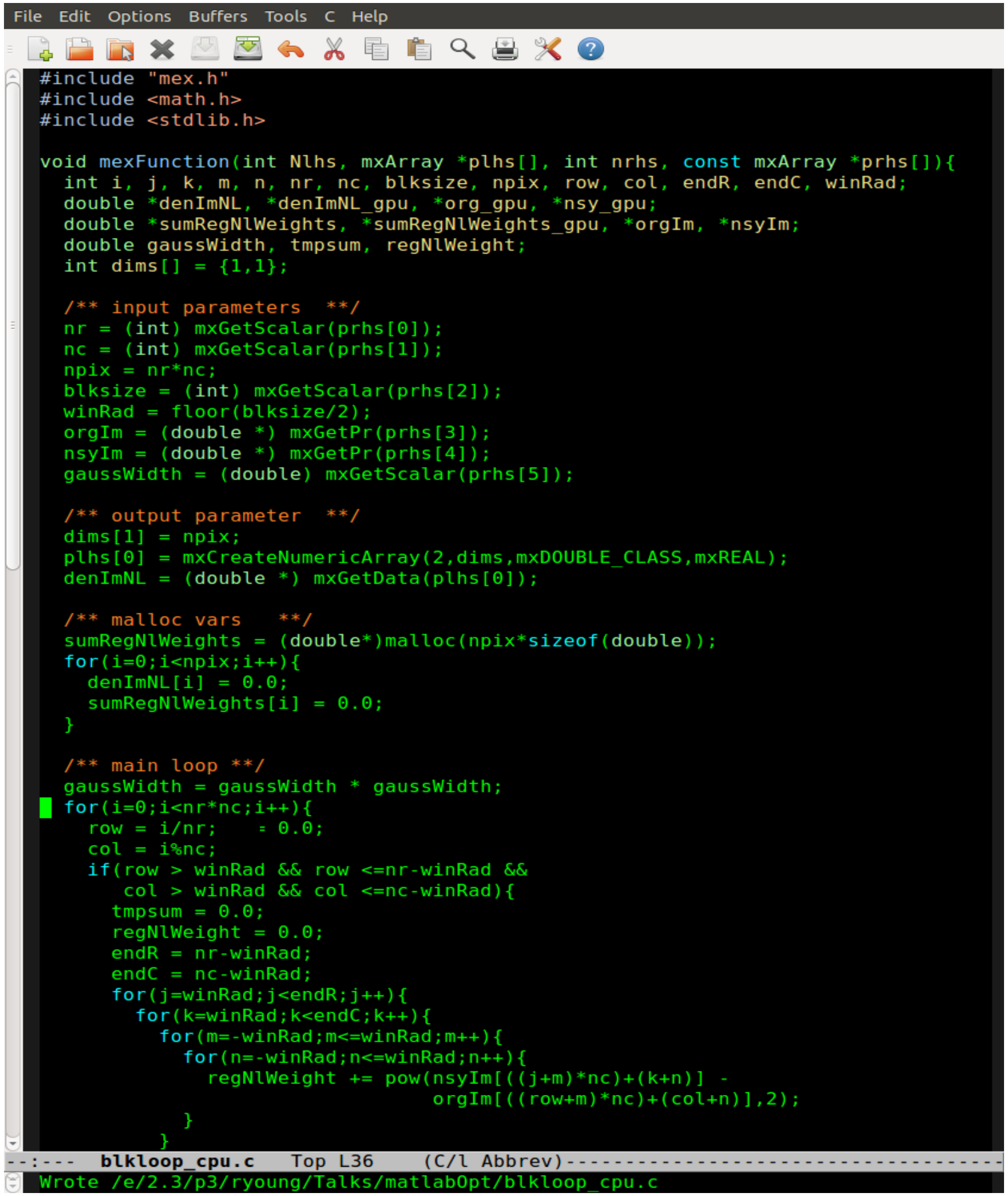
- have a for loop that you can't vectorize?
- if each loop iteration is independent:

```
matlabpool open;
parfor i=1:N
    < loop body >
end
matlabpool close;
```

- current maximum # workers (threads) == 8

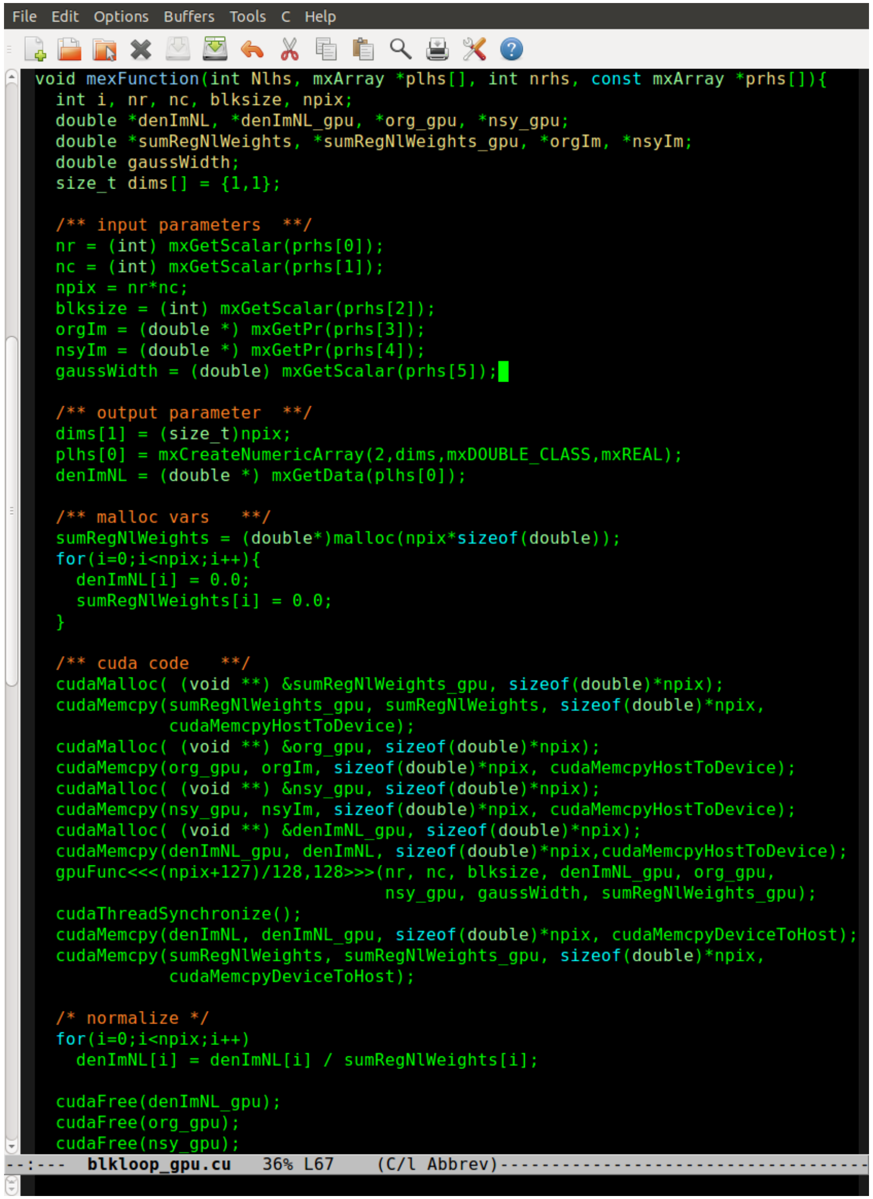
faster code - MEX

- Running C code in Matlab
- Standard C except for matlab interface.

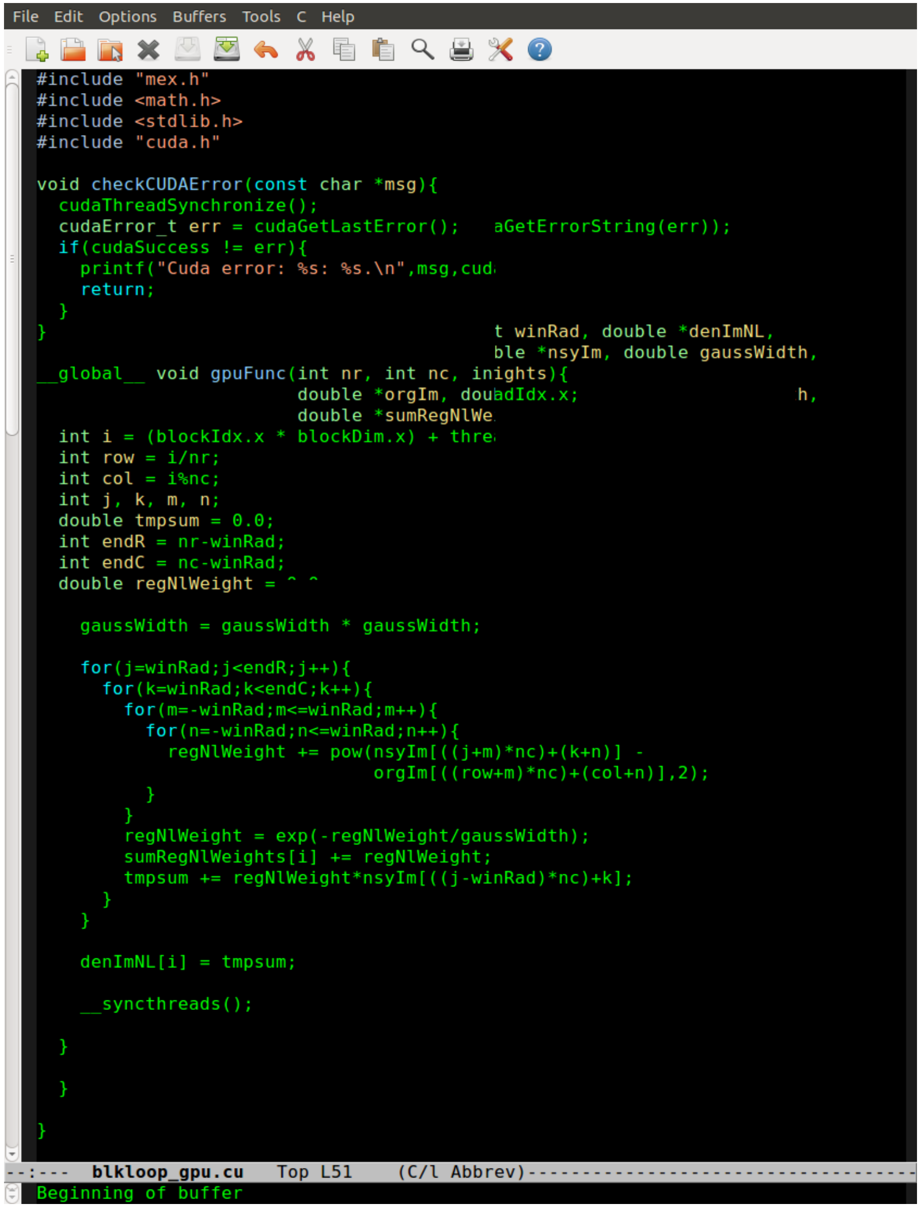


```
#include "mex.h"
#include <math.h>
#include <stdlib.h>

void mexFunction(int Nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]){
  int i, j, k, m, n, nr, nc, blksize, npix, row, col, endR, endC, winRad;
  double *denImNL, *denImNL_gpu, *org_gpu, *nsy_gpu;
  double *sumRegNlWeights, *sumRegNlWeights_gpu, *orgIm, *nsyIm;
  double gaussWidth, tmpsum, regNlWeight;
  int dims[] = {1,1};

  /** input parameters  **/
  nr = (int) mxGetScalar(prhs[0]);
  nc = (int) mxGetScalar(prhs[1]);
  npix = nr*nc;
  blksize = (int) mxGetScalar(prhs[2]);
  winRad = floor(blksize/2);
  orgIm = (double *) mxGetPr(prhs[3]);
  nsyIm = (double *) mxGetPr(prhs[4]);
  gaussWidth = (double) mxGetScalar(prhs[5]);

  /** output parameter  **/
  dims[1] = npix;
  plhs[0] = mxCreateNumericArray(2,dims,mxDOUBLE_CLASS,mxREAL);
  denImNL = (double *) mxGetData(plhs[0]);

  /** malloc vars   **/
  sumRegNlWeights = (double*)malloc(npix*sizeof(double));
  for(i=0;i<npix;i++){
    denImNL[i] = 0.0;
    sumRegNlWeights[i] = 0.0;
  }

  /** main loop **/
  gaussWidth = gaussWidth * gaussWidth;
  for(i=0;i<nr*nc;i++){
    row = i/nr;    = 0.0;
    col = i%nc;
    if(row > winRad && row <=nr-winRad &&
       col > winRad && col <=nc-winRad){
      tmpsum = 0.0;
      regNlWeight = 0.0;
      endR = nr-winRad;
      endC = nc-winRad;
      for(j=winRad;j<endR;j++){
        for(k=winRad;k<endC;k++){
          for(m=-winRad;m<=winRad;m++){
            for(n=-winRad;n<=winRad;n++){
              regNlWeight += pow(nsyIm[((j+m)*nc)+(k+n)] -
                                   orgIm[((row+m)*nc)+(col+n)],2);
            }
          }
--:---  blkloop_cpu.c    Top L36    (C/l Abbrev)------------------------------------
Wrote /e/2.3/p3/ryoung/Talks/matlabOpt/blkloop_cpu.c
```

faster for loops - CUDA



```
void mexFunction(int Nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]){
  int i, nr, nc, blksize, npix;
  double *denImNL, *denImNL_gpu, *org_gpu, *nsy_gpu;
  double *sumRegNlWeights, *sumRegNlWeights_gpu, *orgIm, *nsyIm;
  double gaussWidth;
  size_t dims[] = {1,1};

  /** input parameters  **/
  nr = (int) mxGetScalar(prhs[0]);
  nc = (int) mxGetScalar(prhs[1]);
  npix = nr*nc;
  blksize = (int) mxGetScalar(prhs[2]);
  orgIm = (double *) mxGetPr(prhs[3]);
  nsyIm = (double *) mxGetPr(prhs[4]);
  gaussWidth = (double) mxGetScalar(prhs[5]);

  /** output parameter  **/
  dims[1] = (size_t)npix;
  plhs[0] = mxCreateNumericArray(2,dims,mxDOUBLE_CLASS,mxREAL);
  denImNL = (double *) mxGetData(plhs[0]);

  /** malloc vars   **/
  sumRegNlWeights = (double*)malloc(npix*sizeof(double));
  for(i=0;i<npix;i++){
    denImNL[i] = 0.0;
    sumRegNlWeights[i] = 0.0;
  }

  /** cuda code   **/
  cudaMalloc( (void **) &sumRegNlWeights_gpu, sizeof(double)*npix);
  cudaMemcpy(sumRegNlWeights_gpu, sumRegNlWeights, sizeof(double)*npix,
             cudaMemcpyHostToDevice);
  cudaMalloc( (void **) &org_gpu, sizeof(double)*npix);
  cudaMemcpy(org_gpu, orgIm, sizeof(double)*npix, cudaMemcpyHostToDevice);
  cudaMalloc( (void **) &nsy_gpu, sizeof(double)*npix);
  cudaMemcpy(nsy_gpu, nsyIm, sizeof(double)*npix, cudaMemcpyHostToDevice);
  cudaMalloc( (void **) &denImNL_gpu, sizeof(double)*npix);
  cudaMemcpy(denImNL_gpu, denImNL, sizeof(double)*npix,cudaMemcpyHostToDevice);
  gpuFunc<<<(npix+127)/128,128>>>(nr, nc, blksize, denImNL_gpu, org_gpu,
                                  nsy_gpu, gaussWidth, sumRegNlWeights_gpu);
  cudaThreadSynchronize();
  cudaMemcpy(denImNL, denImNL_gpu, sizeof(double)*npix, cudaMemcpyDeviceToHost);
  cudaMemcpy(sumRegNlWeights, sumRegNlWeights_gpu, sizeof(double)*npix,
             cudaMemcpyDeviceToHost);

  /* normalize */
  for(i=0;i<npix;i++)
    denImNL[i] = denImNL[i] / sumRegNlWeights[i];

  cudaFree(denImNL_gpu);
  cudaFree(org_gpu);
  cudaFree(nsy_gpu);
--:---  blkloop_gpu.cu   36% L67    (C/l Abbrev)--------------------------------
```

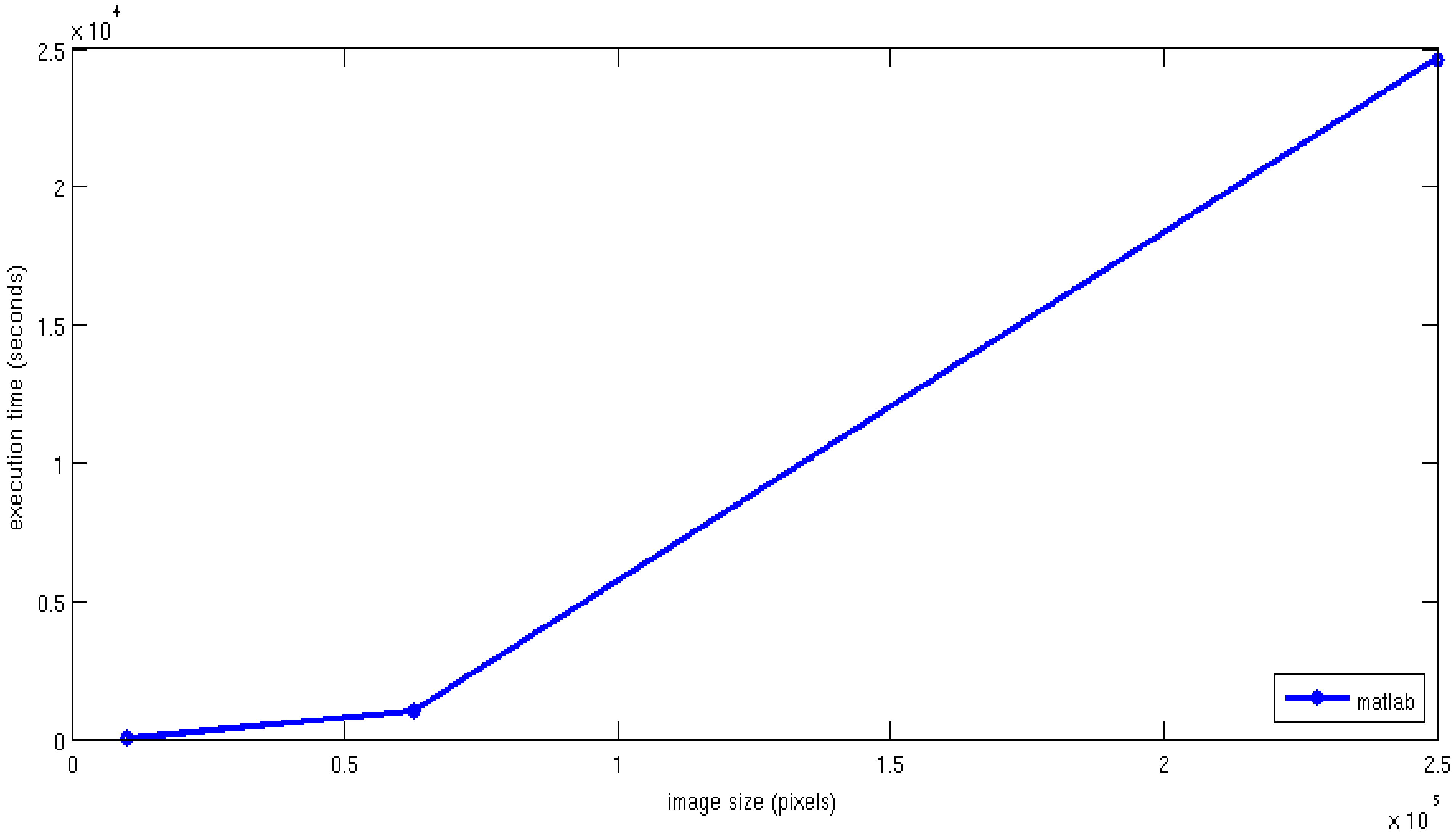


```
#include "mex.h"
#include <math.h>
#include <stdlib.h>
#include "cuda.h"

void checkCUDAError(const char *msg){
  cudaThreadSynchronize();
  cudaError_t err = cudaGetLastError();    aGetErrorString(err));
  if(cudaSuccess != err){
    printf("Cuda error: %s: %s.\n",msg,cud
    return;
  }
}                                          t winRad, double *denImNL,
                                           ble *nsyIm, double gaussWidth,
__global__ void gpuFunc(int nr, int nc, inights){
                        double *orgIm, douadIdx.x;                     h,
                        double *sumRegNlWe
  int i = (blockIdx.x * blockDim.x) + thre
  int row = i/nr;
  int col = i%nc;
  int j, k, m, n;
  double tmpsum = 0.0;
  int endR = nr-winRad;
  int endC = nc-winRad;
  double regNlWeight = ^ ^

    gaussWidth = gaussWidth * gaussWidth;

    for(j=winRad;j<endR;j++){
      for(k=winRad;k<endC;k++){
        for(m=-winRad;m<=winRad;m++){
          for(n=-winRad;n<=winRad;n++){
            regNlWeight += pow(nsyIm[((j+m)*nc)+(k+n)] -
                               orgIm[((row+m)*nc)+(col+n)],2);
          }
        }
        regNlWeight = exp(-regNlWeight/gaussWidth);
        sumRegNlWeights[i] += regNlWeight;
        tmpsum += regNlWeight*nsyIm[((j-winRad)*nc)+k];
      }
    }

    denImNL[i] = tmpsum;

    __syncthreads();

  }

  }

}
--:---  blkloop_gpu.cu    Top L51    (C/l Abbrev)----------------------------------
Beginning of buffer
```

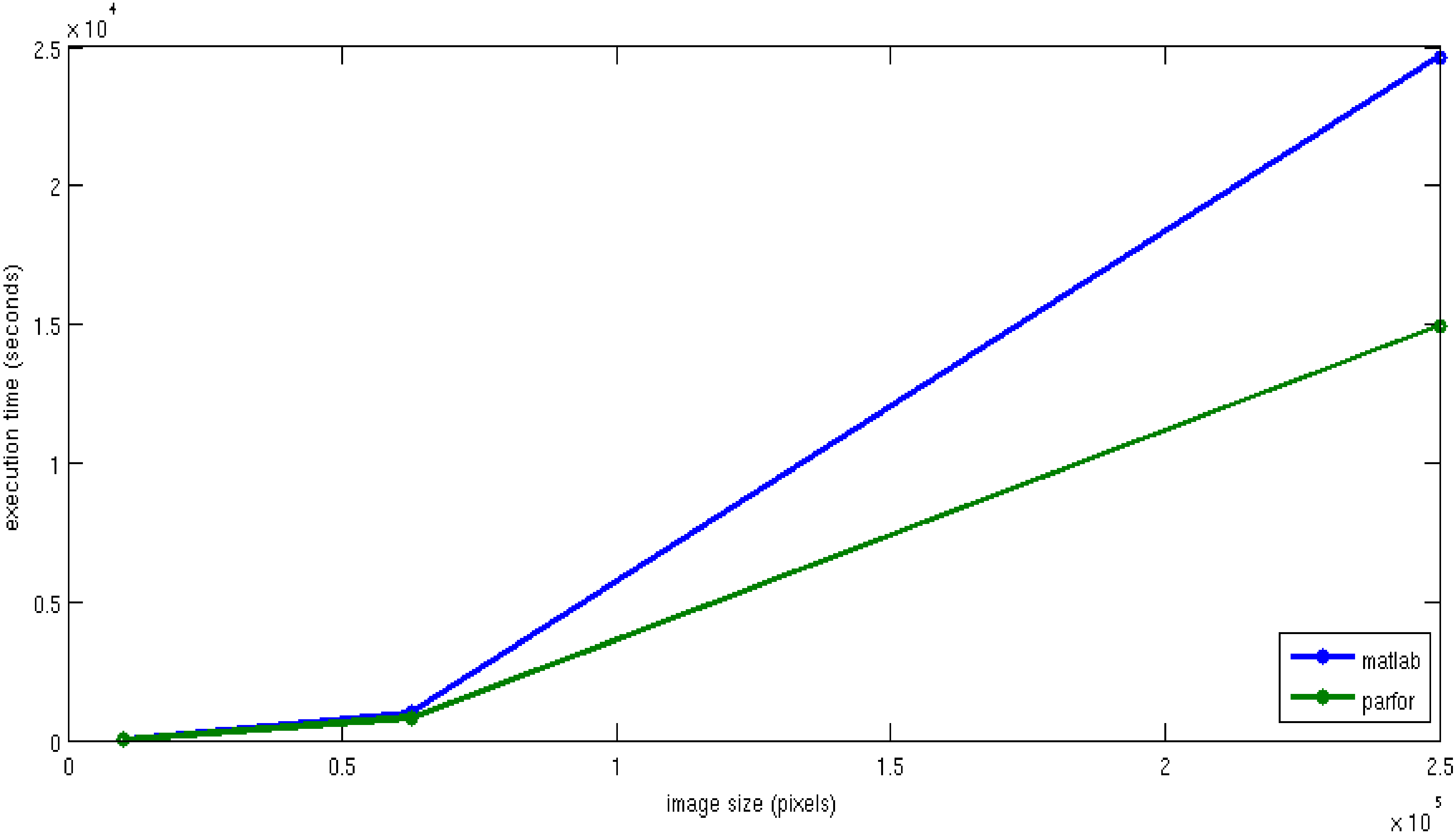
when is CUDA the right answer?

- Loop with large number of iterations
- Few if any temporary variables in loop
  - Large temporary variables must be duplicated
- For example: summary statistics
  - Only memory transfer on to card
  - Small temporary variable
  - Temporary variable can be shared by threads

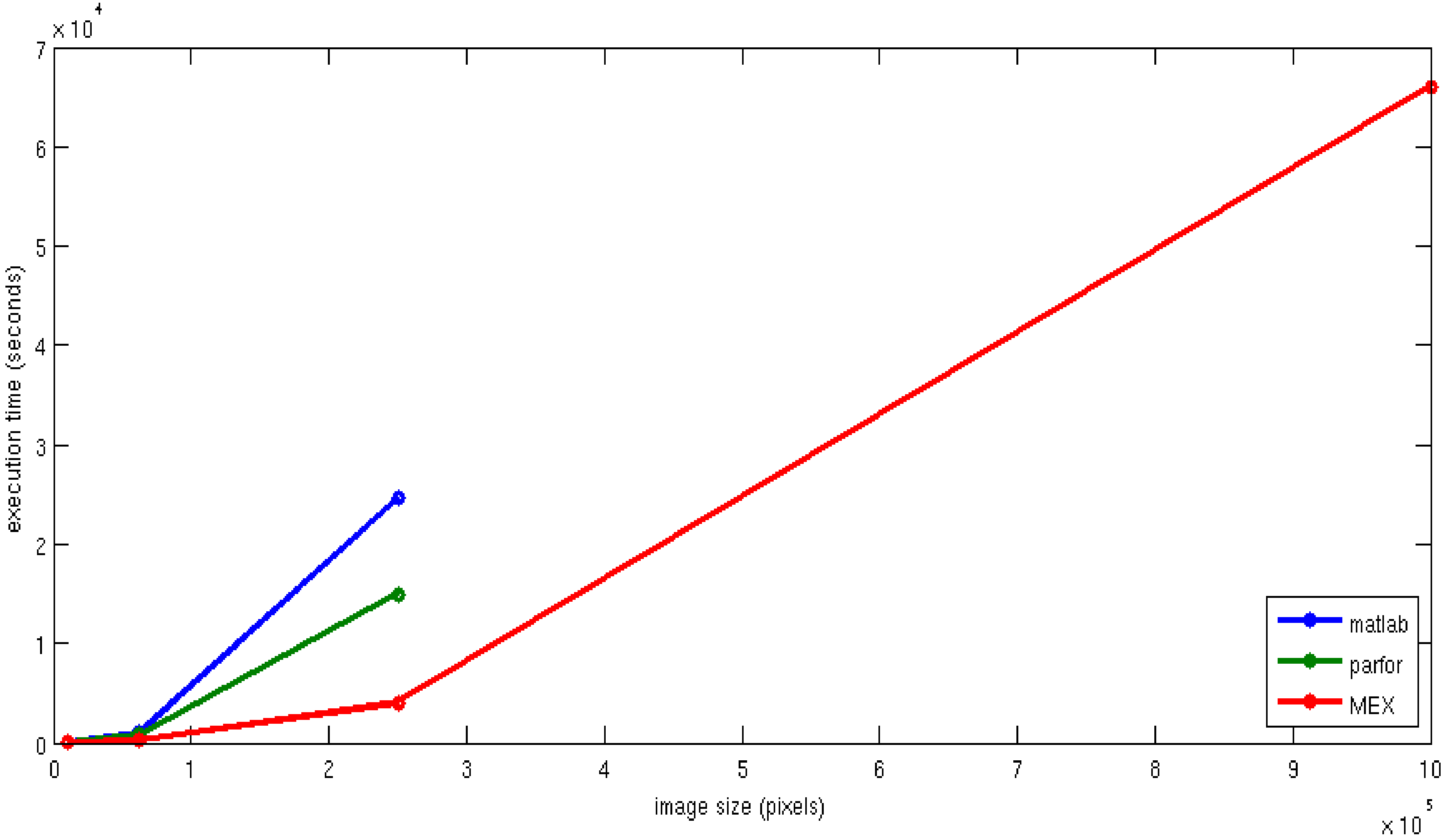
nlmeans speed comparison



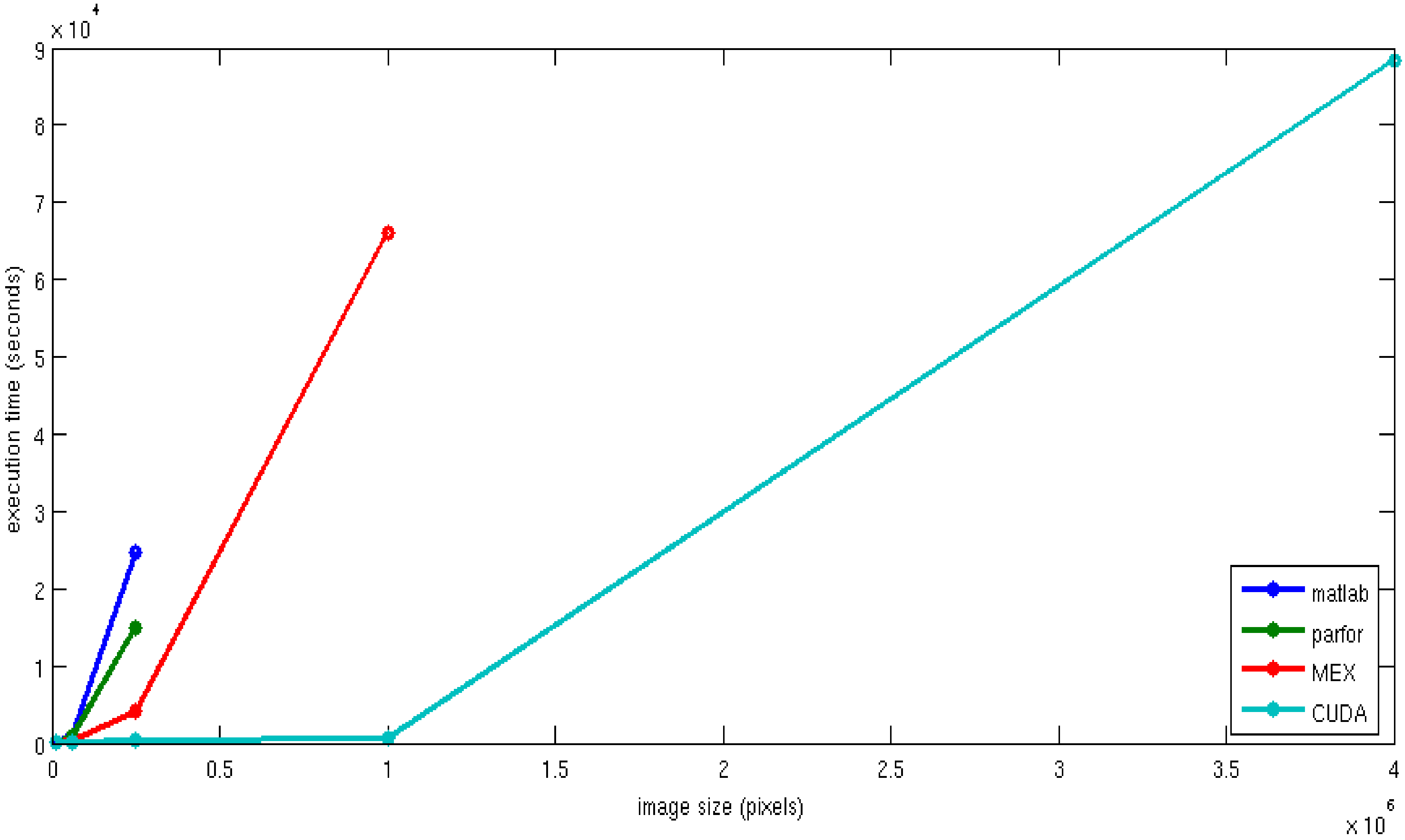
nlmeans speed comparison



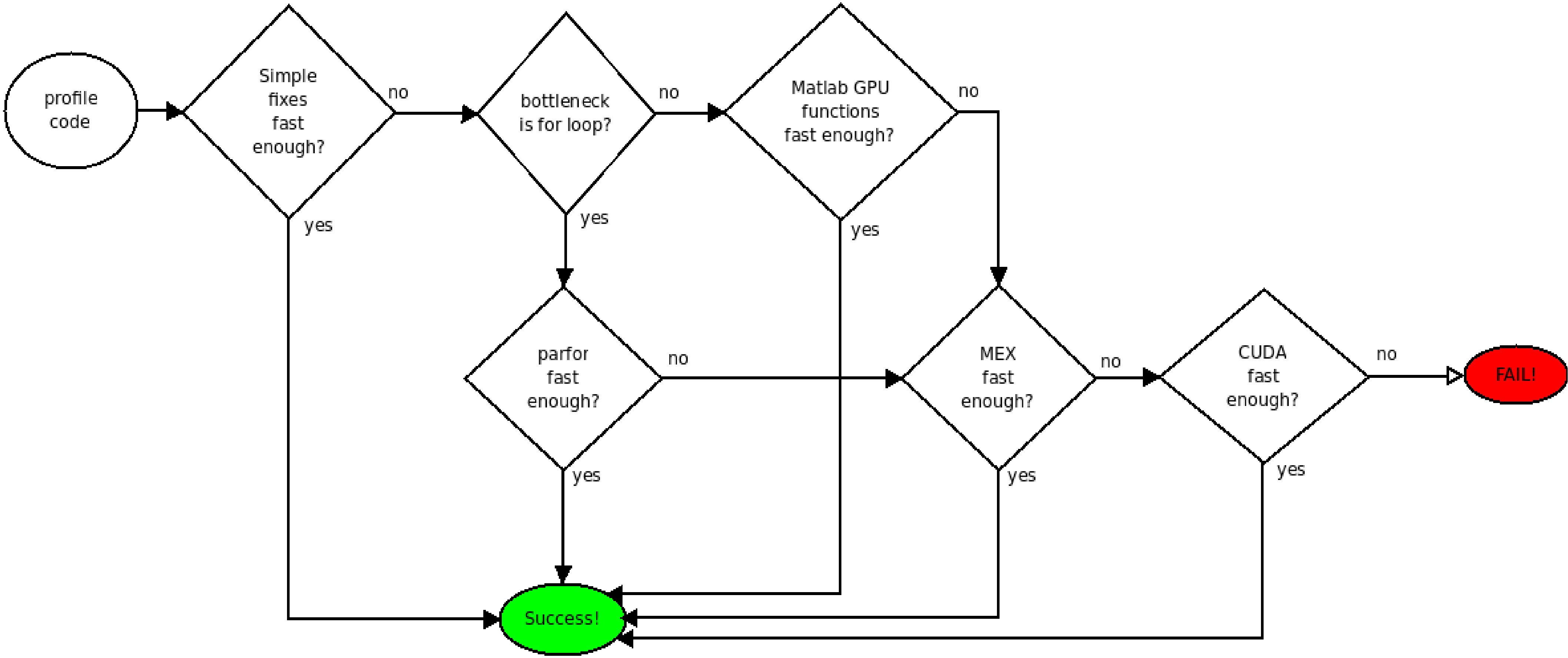
nlmeans speed comparison



nlmeans speed comparison



Summary


profile code
Simple fixes fast enough?
no
yes
bottleneck is for loop?
no
yes
Matlab GPU functions fast enough?
no
yes
parfor fast enough?
no
yes
MEX fast enough?
no
yes
CUDA fast enough?
no
yes
FAIL!
Success!

Resources

- me – my door's always open!
- Matlab blogs (especially Loren & Steve):

  http://blogs.mathworks.com
- general Matlab optimization:

  http://www.mathworks.com/matlabcentral/fileexchange/5685-writing-fast-matlab-code
- profiler:

  http://blogs.mathworks.com/desktop/2010/02/01/speeding-up-your-program-through-profiling/

  http://www.mathworks.com/help/techdoc/matlab_env/f9-17018.html
- parfor:

  http://www.mathworks.com/help/toolbox/distcomp/brb2x2l-1.html

  http://blogs.mathworks.com/loren/2007/10/03/parfor-the-course/
- GPU:

  http://www.mathworks.com/discovery/matlab-gpu.html

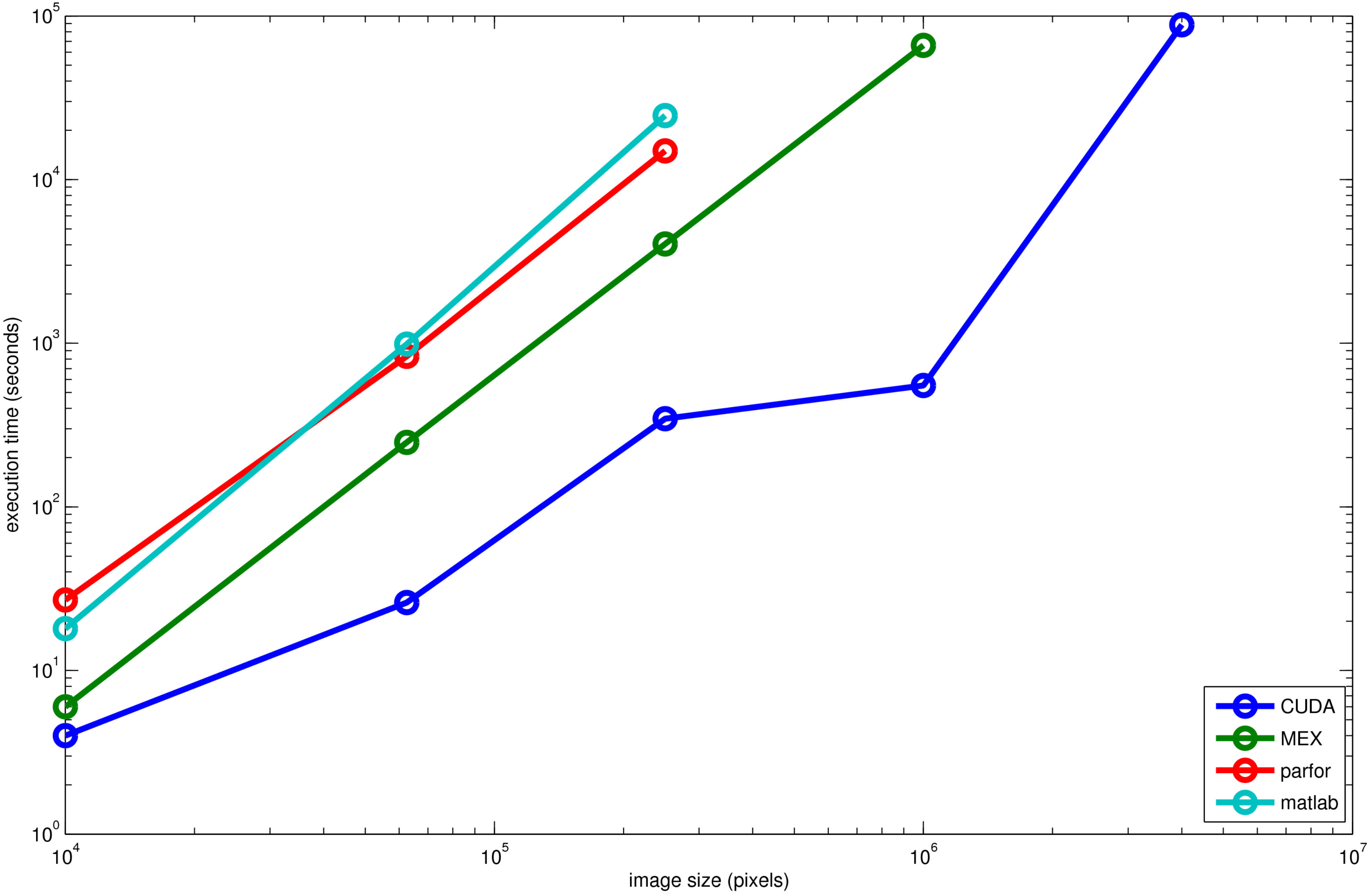
  http://www.mathworks.com/help/toolbox/distcomp/bsic3by.html
- MEX:

  http://www.mathworks.com/support/tech-notes/1600/1605.html

Thanks!

Let's talk about your code!

nlmeans code comparison


10^5
10^4
10^3
10^2
10^1
10^0
execution time (seconds)
10^4
10^5
10^6
10^7
image size (pixels)
CUDA
MEX
parfor
matlab