Improving Matlab Performance

MATLAB and Speed

- MATLAB is an interpreted language
- Each instruction by the user takes time to decode into machine code
- Proper coding techniques can minimize this decode time and maximize program speed
Preallocation of arrays

- MATLAB allows growing arrays in loops, but this is inefficient

```matlab
array1 = [];
array2 = [];
n = 1000;
for i=1:n
    array1 = [array1 5*i] ;
    array2(i) = 5*i ;
end
```

Preallocate, using correct data type

- For double array,
  ```matlab
  array1 = zeros(1,n);
  ```
- For another data type, like int8,
  ```matlab
  array1 = zeros(1,n, 'int8');
  ```
- Do not use
  ```matlab
  array1 = uint8(zeros(1,n));
  ```
- Avoid changing data type of a declared variable
- Instead, have different variable preallocated in correct type
Preallocation demonstrated

Optimizing Functions

- Functions with the same in input vs output arguments operate in-place
  - Standard function declaration:
    - `function y = myfunc (x)`
    - `function [a b c] = myfunc (x)`
  - In-place function declaration
    - `function x = myfunc (x)`
    - `function [x b c] = myfunc (x)`

- Best when only outputs are inputs
- Must call function with same input/output
In-Place Demonstration

Compiling functions

- MATLAB does contain a compiler, which turn MATLAB files into executables
- It allows you to run MATLAB programs on a computer without MATLAB
- In general, doing this will not make your code faster
Vectorization

- An example of for loop code
  
  ```
  i = 0;
  for t = 0:.01:10
    i = i + 1;
    y(i) = sin(t);
  end
  ```

- Vectorized:
  
  ```
  t = 0:.01:10;
  y = sin(t);
  ```

- Most built-in functions accept arrays

Logical Indexing

- Allows simultaneously indexing all values of an array that meet certain logical criterion

- For example, to create an array B which contains all entries in A with value less than 2:
  
  ```
  B = A(A < 2);
  ```

- This is one of the most powerful tools in MATLAB, but can be difficult to learn
Vectorization & Logical Indexing Examples

repmat and reshape

- What if we want to create a matrix whose columns are all of the (x,y) locations in an image?

\[ \text{img} = \text{rand}(3,3); \]

\% want to create the matrix
\% [ 1 2 3 1 2 3 1 2 3; 
\% 1 1 1 2 2 2 3 3 3 ];
repmat and reshape

- What if we want to create a matrix whose columns are all of the (x,y) locations in an image? Could use a nested for loop:

  ```matlab
  [rows, cols] = size(img);
  locs = zeros(2, rows * cols);
  i = 1;
  for y = 1:rows
      for x = 1:cols
          i = i + 1;
          locs(:,i) = [x y]';
      end
  end
  Very slow!
  ```

- Instead can use `repmat`

  ```matlab
  >> help repmat
  repmat Replicate and tile an array.
  B = repmat(A,M,N) creates a large matrix B consisting of an M-by-N tiling of copies of A. The size of B is [size(A,1)*M, size(A,2)*N].
  ```
repmat and reshape

- Instead can use `repmat`
- Replicates a matrix:

```matlab
[rows, cols] = size(img);
colidxs = repmat([1:cols], 1, rows);
% getting the right y coordinates is trickier
rowidxs = reshape(repmat([1:rows], cols, 1), ... 1, rows * cols);
idxs = [ rows; cols ];
% vectorized matlab code can become hard to understand
```

- Not the only (or necessarily the fastest) way to solve this problem

- In Matlab, there are often many ways to solve the same problem
  - Some fast, some slow
  - Some easy to code, others extremely hard
Still not fast enough?

- Use a compiled language
- C/C++ are known for their speed
- C code can be compiled within MATLAB
- This will not be needed for your final projects

Questions?