Lecture 2: Programming in MATLAB

Slides from Padhraic Smyth
Assignment 1

• Assignment 1:
  – MATLAB tutorial
  – BG subtraction on video
  – submit your completed assignment by Thursday 12:30 pm

• Any issues with finding machines with MATLAB?
  – Note only some machines in 364 have MATLAB. See Web page.
  – You can also ask the lab attendant in 364 for help.

• Questions on assignment at the end of the lecture
  – today’s lecture notes may help answering some of your questions
Outline of Today’s Lecture

• Data Types
  – arrays: char, numeric, struct, logical, cell, others

• Operators
  – arithmetic, relational, logical

• Flow Control
  – conditionals, case, while, etc.

• M-functions
  – syntax
  – examples of simple functions
  – writing and debugging a simple MATLAB function

• Next lecture:
  – Classification algorithms
Data Types in MATLAB

Array

- Logical
- Char ‘a’
- Numeric
  - Uint8 (8 bit, from 0 to 255, e.g., pixel values)
  - Double e.g., 3.2567 (8 bytes)
- Structure
  - image.width = 120
  - image.name = ‘face1’

Others
Uint8 and Doubles

- Double
  - almost all MATLAB functions
    - expect doubles as arguments
    - return doubles
Uint8 and Doubles

- Double
  - almost all MATLAB functions
    - expect doubles as arguments
    - return doubles
Uint8 and Doubles

- Double
  - almost all MATLAB functions
    - expect doubles as arguments
    - return doubles
    - e.g.,

  • need to convert uint8 to double before performing any math operations

```matlab
» a = 1:10
a =
1 2 3 4 5 6 7 8 9 10
» b = uint8(a)
b =
1 2 3 4 5 6 7 8 9 10
» whos
Name      Size         Bytes  Class
a         1x10            80  double array
b         1x10            10  uint8 array

» b*500
255 255 255 255 255 255 255 255 255 255

» double(b)*500
ans =
500 1000 1500 2000 2500 3000 3500 4000 4500 5000
```
**Plotting an image**

- Use imshow.m

```matlab
>> help imshow
```

IMSHOW(I) displays the grayscale image I

IMSHOW(RGB)
  - if RGB is of type uint8, it assumes 0=dark and 255 = bright
  - if RGB is of type double, it assumes 0=dark and 1 = bright

IMSHOW(BW) displays the binary image BW. IMSHOW display pixels with the value 0 (zero) as black and pixels with the value 1 as white.

IMSHOW(X,MAP) displays the indexed image X with the colormap MAP. X takes on values from 1 to “k”, and colormap is a matrix of size “k by 3”

- Requires java virtual machine. I tend to use image.m and imagesc.m which do not.
Char Data Type

```matlab
» c = ['hello'];

» whos

Name      Size          Bytes        Class
c              1x5             10          char array

Grand total is 5 elements using 10 bytes

» c(1)
ans = h
```
Char Data Type

```matlab
» c = ['hello'];

» whos
Name      Size          Bytes        Class
     c              1x5             10          char array

   Grand total is 5 elements using 10 bytes

» c(1)
ans = h
```

```matlab
» d = [c,' again'];
» d
d = hello again
»
» b = ['hello';'again'];
» size(b)
ans = 2     5
» b
b =
    hello
    again
»
```
Char Data Type

```matlab
» c = ['hello'];

» whos
   Name      Size          Bytes        Class
   c              1x5             10          char array

   Grand total is 5 elements using 10 bytes

» c(1)
   ans = h

» d = [c,' again'];
» d
   d = hello again
   »
» b = ['hello';'again'];
» size(b)
   ans = 2    5
» b
   b =
       hello
       again
   »

Many string functions available, e.g., strcat, num2str, etc
```
Struct Data Type

im.index = [4 10 3; 12 12 2; 10 4 3];
Struct Data Type

>> im.index = [8 10 2; 22 7 22; 2 4 7];
>> im.map = [0 0 0; 0 0 .1; 0 .1 0;.....]

» whos
    Name    Size      Bytes  Class
      im    1x1        590  struct

Grand total is 18 elements using 590 bytes

This is how gif images are stored internally
Arrays of Structures

```matlab
» ims(1) = im;
» ims(2).index = [12 3 2; 23 3 3; 23 12 1];
» ims(2).cmap  = ims(1).map;
» whos
    Name      Size         Bytes  Class
    __________  __________    _______  __________
    ims       1x2            894  struct array

Grand total is 28 elements using 894 bytes

» ims

ims =

1x2 struct array with fields:
    index
    map
```
Arrays of Structures

» ims(1) = im;
» ims(2).index = [12 3 2; 23 3 3; 23 12 1];
» ims(2).cmap = ims(1).map;
» whos
  Name      Size     Bytes  Class       Attributes
  image     1x2         894  struct array

Grand total is 28 elements using 894 bytes

» ims

image =

1x2 struct array with fields:
  index
  cmap

» ims(2)

ans =

  index: [3x3 double]
  map: [27x3 double]

» ims(1)

ans =

  index: [3x3 double]
  map: [27x3 double]
Array of structures (example)

>> filelist = dir('*.jpg');
>> filelist

filelist =

29x1 struct array with fields:
  name
  date
  bytes
  isdir
  datenum

>> filelist(1)

ans =

  name: 'aam.jpg'
  date: '27-Feb-2010 13:36:11'
  bytes: 3668
  isdir: 0
  datenum: 7.3420e+05

>> im = imread(filelist(1).name);

Useful for enumerating all frames in a directory (eg, homeworks)
Operators

- Arithmetic
  - numeric computations, e.g., $2^{10}$

- Relational
  - quantitative comparison of operands
  - e.g., $a < b$

- Logical
  - AND, OR, NOT, etc
  - result of type Logical, 1 (TRUE) or 0 (FALSE)
Arithmetic Operators

- Transpose, $a'$

- Power, $a^2$

- Addition, multiplication, division
  - $a(1)\times b(2)$
  - $a\times b$
    - works if $a$ and $b$ are matrices with appropriate dimensions ($\text{columns}(a) = \text{rows}(b)$)
  - $a.*b$ (element by element)

- except for matrix operations, most operands must be of the same size, unless one is a scalar
Arithmetic Operators

- Transpose, $a'$

- Power, $a^2$

- Addition, multiplication, division
  - $a(1) \cdot b(2)$
  - $a \cdot b$
    - works if $a$ and $b$ are matrices with appropriate dimensions ($\text{columns}(a) = \text{rows}(b)$)
  - $a \cdot * b$ (element by element)

- except for matrix operations, most operands must be of the same size, unless one is a scalar

```matlab
» a = [2 3];
» b = [4 5];
» a(1) * b(2)
ans =
    10

» a * b
??? Error using ==> *
Inner matrix dimensions must agree.

» a * b'
ans =
    23

» a.*b
ans =
    8    15

» b/2
ans =
    2.0000    2.5000
```
Relational Operators

- `<`, `<=`, `>`, `>=`, `==`, `~=`

- compare corresponding elements of arrays with same dimensions

- if one is scalar, one is not, the scalar is compared with each element

- result is of type Logical
  - element by element 1 or 0
Relational Operators

- `<`, `<=`, `>`, `>=`, `==`, `~=`
- compare corresponding elements of arrays with same dimensions
- if one is scalar, one is not, the scalar is compared with each element
- result is of type Logical
  - element by element 1 or 0

```plaintext
» a
a = 2  3

» b
b = 4  5

» a > b
ans = 0  0

» b > a
ans = 1  1

» a > 2
ans = 0  1
```
Flow Control

- If, else, endif
  
  if index<100
  statements
  else
  statements
  end

- For.....
  For i = 1:100
  statements
  end

- Switch, while, case, etc
Vectorization of Computation

tic
for i=1:100000
    y(i) = log(i);
end
toc
Vectorization of Computation

tic
for i=1:100000
    y(i) = log(i);
end
toc

elapsed_time = 168.78 seconds
Vectorization of Computation

```matlab
tic
for i=1:100000
    y(i) = log(i);
end
toc

elapsed_time = 168.78 seconds
```

```matlab
tic
i=1:100000;
z = log(i);
toc
```

Vectorization of Computation

First method calls the log function 100,000 times,
Second method only calls it once (much faster)
Memory Preallocation

- What happens in the previous example if we preallocate memory to $y$ and $z$? e.g.,

```plaintext
y = zeros(10000,1);
z = zeros(10000,1);
```
MATLAB Programming

- “M File” = text file with MATLAB code, e.g., sort.m

- Two kinds of M-files
  - scripts
    - no input arguments supplied
    - no output arguments returned
    - operates on data in workspace
  - functions
    - can accept input arguments and return output arguments
    - internal variables local to function by default
    - useful for extending functionality of MATLAB
Example of a MATLAB script

% script randscript
% A simple script to generate a vector of n
% random numbers. We calculate the numbers
% using (a) for loops, and (b) a direct function call.
%
% Professor Smyth, Oct 2007

Comment text: it is always important to put comments in your code. Comments at the top should clearly explain what the script or function does
Example of a MATLAB script

% script randsum
% A simple script to generate a vector of n
% random numbers. We calculate the numbers
% using (a) for loops, and (b) a direct function call.
%
% Professor Smyth, Oct 2007

n = 100000; % the number of points for the "for loop"
y = zeros(n,1); % preallocate memory for y
fprintf('Simulating %d random numbers.....\n\n',n);

Initialize various variables
Print out some information to the screen
Example of a MATLAB script

% script randsum
% A simple script to generate a vector of n
% random numbers. We calculate the numbers
% using (a) for loops, and (b) a direct function call.
%
% Professor Smyth, Oct 2007

n = 100000; % the number of points for the "for loop"
y = zeros(n,1); % preallocate memory for y
fprintf('Simulating %d random numbers.....\n\n',n);

% first do the calculation using a "for loop"
fprintf('For loop calculations.....\n');
tic % set the timer
for i=1:n
    y(i) = rand(1);
end
total = sum(y);
fprintf('Sum of %d random numbers = %f\n',n,total);
t1 = toc; % read the time elapsed since "tic" (in seconds)
fprintf('Time taken, using for loop = %6.5f microseconds\n\n', (t1)*1000);

......

(1) Calculate the n random numbers and their sum using a for loop,
(2) record the time taken, and (3) print to the screen
Example of a MATLAB script

% now do the calculation using vectorization
fprintf('Vectorization calculations.....\n');

 tic % reset the timer
 z = rand(n,1);
 total = sum(z);

 fprintf('Sum of %d random numbers = %f
',n,total);
 t2 = toc; % read the time elapsed since "tic" (in seconds)
 fprintf('Time taken, using vectorization = %6.5f microseconds
', (t2)*1000);

(1) Now calculate n random numbers
and their sum using a direct function call
(2) record the time taken,
and (3) print to the screen
Example of a MATLAB function

```
function [sum, difference] = sumdiff(a, b);
```

- **Tells MATLAB this is a function**
- **List of output values returned (can be any form of array data type)**
- **Name of the function**
- **List of input argument values, comma delimited (any form of array data type)**
Example of a MATLAB function

```matlab
function [sum, difference] = sumdiff(a, b);
% function [sum, difference] = sumdiff(a, b);
%
% A simple function to compute the sum and difference
% of two input arguments a and b
%
% Professor Smyth, Oct 2007
%
% INPUTS:
%    a: array of size r x c
%    b: array of size r x c
%
% OUTPUTS:
%    sum: a + b
%    difference: a - b
```

Clear comments in function headers are very useful

Note the explicit statement explaining what the inputs and outputs are (including their dimensionality)
Example of a MATLAB function

function [sum, difference] = sumdiff(a, b);
% function [sum, difference] = sumdiff(a, b);
% % A simple function to compute the sum and difference
% % of two input arguments a and b
% %
% Professor Smyth, Oct 2007
%

% error checking
[rowsa, colsa] = size(a);
[rowsb, colsb] = size(b);
if ( rowsa ~= rowsb ) | ( colsa ~= colsb)
    error('sizes of a and b do not match');
end

Error checking is always a good idea!
function [sum, difference] = sumdiff(a, b);
% A simple function to compute the sum and difference
% of two input arguments a and b
% Professor Smyth, Oct 2007
%
% error checking
[rowsa, colsa] = size(a);
[rowsb, colsb] = size(b);
if ( rowsa ~= rowsb ) | ( colsa ~= colsb)
    error('sizes of a and b do not match');
end

sum = a + b;
difference = a - b;

Finally, the actual computational part of the function
MATLAB functions in general

- Function line definition
  - required of all functions

- List of inputs and outputs
  - comma delimited: \[y, z\] = average(a, b, c)
  - for more than one output, outputs are enclosed in square brackets

- Input variables
  - variables within function are local to the function
  - input variables are readable to the function, but not writable
  - values of returned arguments are passed back

- Search path
  - MATLAB searches in this order:
    - variable name, subfunction, current directory, MATLAB search path