Administrivia

My name = Patricia Thaine
My email = pthaine@cs.toronto.edu

Based on slides by Frank Rudzicz
Agenda

- Matlab Basics
- Matrices and vectors
- Displaying results
- Cell arrays and structures
- Miscellaneous topic
MATrix LABoratory

- Designed for matrix computations.
- High-level language
  - You don't see the details
  - Dynamic typing
- Basic Data type
  - 2- or 3-D double-precision matrix
- Most functions work on entire matrices.
MATrix LABoratory

Comprehensive and optimized toolboxes and libraries

  e.g., statistics and neural networks
Useful for prototyping and scientific work, but non-free and not arbitrarily flexible.

Available on cdf:  greywolf:~% matlab

To use version with graphics from your own computer:  ssh -X yourcdfaccount
Getting Help


Within matlab (>>help function-name):

```matlab
>>help log
LOG    Natural logarithm.
    LOG(X) is the natural logarithm of the elements of X.
    Complex results are produced if X is not positive.

See also log1p, log2, log10, exp, logm, reallog.
```
Matrices in Matlab

- Generate a “magic square” with equal row, column, and diagonal sums.
- Assign the resulting matrix to variable A.

```
>> A = magic(3);
>> A = magic(3)
A =
     8     1     6
     3     5     7
    4     9     2
```

No semi-colon, so its result is shown. Useful for debugging, but NOT to be handed in.
Matrices in Matlab

Colon generates number sequence:

```
>> 11:14
ans =
   11   12   13   14
>> -1:1
ans =
   -1    0    1
>> 3:0
ans =
Empty matrix: 1-by-0
```

Second colon specifies step size:

```
>> 1:3:12
ans =
     1     4     7    10
>> 4:-1:1
ans =
     4     3     2     1
>> 3:-0.5:2
ans =
    3.0000    2.5000    2.0000
```
Matrices in Matlab

```matlab
>> A = magic(3)
A =
     8     1     6
     3     5     7
     4     9     2

Select submatrices

>> A(2,1)
an =
     3

>> A(1,:)
an =
     8     1     6

>> A(2:3,1:2)
an =
     3     5
     4     9

>> A(1:5)
an =
     8     3     4     1     5

>> b=A(6:end)
an =
     9     6     7     2

>> size(b)
an =
     1     4
```
Matrices in Matlab

```matlab
>> C = [2 7; 3 1]
C =
    2  7
    3  1

>> D = [A(:, end) A(1,:)']
D =
    6  8
    7  1
    2  6
```

Build manually

Transpose of a matrix

```matlab
>> A = magic(3)
A =
    8  1  6
    3  5  7
    4  9  2
```

```matlab
>> [zeros(1,3); A(2,:)]
ans =
    0  0  0
    3  5  7
```

zeros(1,3)
Mask matrix elements

```matlab
>> find(A>5)
ans =
   1
   6
   7
   8

>> A(find(A>5)) = 0
A =
   0   1   0
   3   5   0
   4   0   2
```

```matlab
>> A = magic(3)
A =
   8   1   6
   3   5   7
   4   9   2
```
Operators preceded by a period (.) perform the given operation in an *element-wise* fashion.
Reporting in Matlab: disp

>> a = 5;
>> disp(a);
   5
>> a = [5 6 7];
>> disp(a);
   5   6   7
>> str = 'Array';
>> disp([str, ': ', num2str(a)]);
Array: 5  6  7

The disp command will be your primary way of reporting results. Leaving semicolon adds variable name, not always desirable.
Reporting in Matlab: plot

\[
\begin{align*}
    & t = 0:0.1:10; \\
    & x = \exp(t \times (j - 1/3)); \\
    & \text{plot}(t, \text{real}(x), t, \text{imag}(x)); \\
    & \text{grid}; \\
    & \text{legend}('\text{real}', '\text{imaginary}'); \\
    & \text{saveas(gcf, 'realImag.eps');}
\end{align*}
\]

\[
\begin{align*}
    & x = 0:20; \\
    & y = 0.5 - 0.5\cos(2\pi x/20); \\
    & \text{stem}(x, y); \\
    & \text{title}('20-point raised cosine');
\end{align*}
\]
Good practice: vectorizing

Loops are slow in Matlab.

Matrix operations are very fast.

e.g., compute logarithms for elements of a matrix

BAD:

```matlab
P = randn(50,100);
for i=1:size(P,1)
    for j=1:size(P,2)
        P(i,j) = log( P(i,j) );
    end
end
```
Good practice: vectorizing

Loops are slow in Matlab.

Matrix operations are very fast.

e.g., compute logarithms for elements of a matrix

**BAD:**

```matlab
P = randn(50,100);
for i=1:size(P,1)
    for j=1:size(P,2)
        P(i,j) = log( P(i,j) );
    end
end
```

**GOOD:**

```matlab
P = randn(50, 100);
P = log(P);
```
Some equations are secretly vector operations in disguise.

e.g., compute

\[ \sum_{n=1}^{1000} n(1000-n) \]

**BAD:**

```matlab
ssum = 0;
for n=1:1000
    ssum = ssum + n*(1000-n);
end
```
Some equations are secretly vector operations in disguise.

e.g., compute

\[ \sum_{n=1}^{1000} n(1000-n) \]

**BAD:**

```matlab
ssum = 0;
for n=1:1000
    ssum = ssum + n*(1000-n);
end
```

**GOOD:**

```matlab
a = 1:1000;
b = 1000 - a;
ssum = a*b';
```
Matlab Miscellany: Cell Arrays

>> C = cell(2,3)
C =
    []     []     []
    []     []     []

>> C{1,2} = 'hello';
>> C{1,1} = eye(2);
>> C{2,2} = 42
C =
    [2x2 double]    'hello'     []
    []             [   42]     []

>> C{1,1}
an =
    1     0
    0     1

>> C(1,1)
an =
    [2x2 double]
Can act similarly to C's struct or to a dictionary.

```matlab
>> s.code = 401;
>> s.name = 'natural language computing';
>> s
s =
    code: 401
    name: 'natural language computing'
>> isfield(s, 'code')
ans =
    1
>> f = fieldnames(s)
f =
    'code'
    'name'
>> s('code')
ans =
    401
>> s.(f{2})
ans =
    natural language computing
>> s(2).code=777
>> s(2).name='NLP'
>> s(2)
ans =
    code: 777
    name: 'NLP'
```
Matlab Miscellany: Directories

The dir function gives a structure describing a directory.

```matlab
>> D = dir('./*PHN')
D =
3x1 struct array with fields:
    name
    date
    bytes
    isdir
    datenum

>> D(1).name
ans =
    SX127.PHN
```
Matlab Miscellany: Files

>> type SX127.PHN
0 2231 h#
2231 2834 dh
2834 3757 iy
...
>> [t1, t2, phone] =
textrd('SX127.PHN', '%d %d %s');
>> t1(2)
ans =
    2231
>> t2(2)
ans =
    2834
>> phone(2)
ans =
    'dh'

>> mfcc = dlmread('unkn_14.mfcc');
>> size(mfcc)
ans =
    279    14

The textread and dlmread functions quickly parse files.
Regular Expressions

Pattern Matching
>> t=regexp('This assignment is awesome!', '([^*]is', 'tokens');
>> t{1}
ans =
   'Th'

Replace String
>> t =regexprep('This assignment is awesome!', '([^*]is', 'at')
t =
at assignment at awesome!
Creating a Function and a File

myfun.m

%Comment: \([y_1, \ldots, y_N]\) are the outputs,
% **myfun** is the function name
%(x_1, \ldots, x_M) are the inputs

function [y_1, \ldots, y_N] = myfun(x_1, \ldots, x_M)
  y_1 = prod(x_1);
  ...
  y_N = prod(x_N);
end

- It is good practice to name the matlab file the same name as the first (and usually the only) function in it

- You may create more than one function within a file, but any function other than the first will only be accessible from within their own file

- You can call 'myfun' from the matlab terminal or from within another file by doing, for example:
  \(\texttt{myfun(var}_1, \ldots, \texttt{var}_M)\)
Matlab Cheatsheet

*, ^
% matrix multiply, exponent

/\, \, inv
%A/B = AB-1, A\B = A-1B, A-1

+, -, .*, ./, .^
% element-wise operations

==, ~=, <, >, <=, >=
% relations

length, size
% size of vectors, matrices
(row-first order)

zeros, ones, eye, diag
% all-0, all-1, identity, diagonal matrices

xlabel, ylabel, zlabel
% label plot axes

subplot(i,j,k)
% plot to the k\’th spot of a ixj array of plots.

sum, prod, min, max
% aggregate rows, columns

find
% list non-zero indices

figure, saveas
% open, save figures

whos
% list name, size, and type of all workspace variables