MATLAB
The Language of Technical Computing

ELEG3124
System & Signal Analysis Laboratory

Farhad Parsan
Fall 2011
What’s MATLAB?

- **MATLAB (MATrix LABoratory)**
- A numerical & symbolic computing environment and a programming language
- Developed by MathWorks Inc.
What can MATLAB do?

• matrix manipulations
• plotting of data and functions
• implementation of algorithms
• interfacing with programs written in other languages, including C, C++, Java, and Fortran
Where is MATLAB used?

- Engineering
- Science
- Economics

MATLAB has a very rich collection of functions used in numerous applications

MATLAB is widely used in academic and research institutions as well as industrial enterprises
Running MATLAB
```
>> a = [1,2,3]

a =
    1     2     3

>> b=[1;2;3]

b =
    1
    2
    3

>> c=a*b

c =
    14
```

Vectors/Matrices

• MATLAB treats all variables as matrices
• Vectors are special forms of matrices and contain only one row OR one column
• Scalars are matrices with only one row AND one column
Vectors/Matrices Examples

- 1
- 1+2
- \(a = [1 2 3]\)
- \(b = [4,5,6]\)
- \(c = a+b\)
- \(d = a+b;\)
- \(d\)
- \(\text{who}\)
- \(\text{whos}\)
- \(\text{clear d}\)
- \(\text{clear all}\)
Vectors/Matrices

• **ans** is a special variable in MATLAB which always stores the result of an expression if it’s not explicitly assigned to a variable

• The answer will not be displayed when a semicolon (;) is put at the end of an expression

• **who** shows which variables are already defined

• **whos** shows more details than who

• **clear variable** clears the specified variable from the workspace

• **clear all** clears all the variables from the workspace
Vectors/Matrices Algebra

- \( a = [1 \ 2 \ 3] \)
- \( b = 2 \times a \)
- \( c = a/0.5 \)
- \( d = a - 1 \)
- \( \Pi \)
- \( i \)
- \( j \)
- `save myvars a b`
- `clear all`
- `whos`
- `load myvars a b`

+ addition
- subtraction
* multiplication
/ division
Vectors/Matrices

• Some variable such as pi, i, j, ... are predefined in MATLAB

• **save filename variables** saves the specified variables in a file named filename in the working directory

• **load filename variables** loads the selected variables from filename

• for save and load command if variables are not specified then all the variables in the workspace will be saved/loaded
Vectors/Matrices Algebra

- $a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
- $b = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$
- $c = a + b$
- $d = a - b$
- $a * b$
- $a .* b$
- $a / b$
- $a ./ b$

+ addition
- subtraction
.* multiplication
* multiplication
./ division
/ division
Vectors/Matrices

• Period (.) before an operation tells MATLAB to do it on individual elements than the whole matrix. For example, A*B multiplies matrix A to matrix B but A.*B multiplies each element of matrix A to the similar element of matrix B.

• For addition and subtraction, both type of operations are essentially the same so there is no need to make a distinction therefore no period is allowed to be used before + or -.
Vectors/Matrices Manipulation

- $a(1,1)$
- $a(1,1) = 5$
- $a(1,:)$
- $a(:,1)$
- $a(1:2,1:2)$
- $a(1,:) = [2,2,2]$
- $b = 0:5$
- $b = 0:2:10$
- $c = \text{linspace}(0,10,6)$
- $c = \text{logspace}(1,4,4)$
- clc
Vectors/Matrices

• Matrix (vector) indexes start from 1
• i:j creates a vector starting from i to j
• i:k:j creates a vector starting from i to j by increments of k
• linspace(i,j,n) creates n linearly spaced points between i and j
• logspace(i,j,n) creates n linearly spaced points between $10^i$ and $10^j$
• clc command clears the screen
Special Matrices

- `ones(i,j)` creates an $i \times j$ matrix in which all the elements are 1
- `zeros(i,j)` creates an $i \times j$ matrix in which all the elements are 0
- `rand(i,j)` creates an $i \times j$ matrix in which the elements are random numbers between 0 and 1
- `eye(i)` creates an $i \times i$ identity matrix
- `magic(i)` creates an $i \times i$ magic matrix
- `det(A)` calculates the determinant of $A$
- `inv(A)` calculates the inverse of matrix $A$
- $A'$ gives the transpose of $A$
- `[A B]` concatenates $A$ to $B$
- `length(A)` gives the length of a vector
- `size(A)` gives the size of a matrix
Plotting

- `plot(x,y)` Plot of x versus y.
- `plot(y)` Plots columns of y versus their index.
- `plot(x,y,'s')` Plots x versus y according to rules outlined by s.
- `grid on/off` Adds/removes grid to current figure.
- `title('text')` Adds title text to current figure.
- `xlabel('text')` Adds x-axis label text to current figure.
- `ylabel('text')` Adds y-axis label text to current figure.
- `hold on/off` Holds current figure as is so subsequent plotting commands add to existing graph.
Plotting example

- \( x = -\pi:.1:\pi; \)
- \( y = \sin(x); \)
- \( \text{plot}(x,y) \)
- grid on
- \( \text{xlabel('X')} \)
- \( \text{ylabel('Y')} \)
- \( \text{title('sine wave')} \)
- \( z = \cos(x); \)
- \( \text{plot}(x,z,'r*') \)
- hold on
- \( \text{plot}(x,y,'b-.') \)
More Plotting Commands

- `clf` clears the current figure
- `stem(x,y)` plot of x versus y for discrete data
- `figure(k)` creates a new figure with number k
- `close(k)` closes the figure with number k
- `subplot(mnk)` divides the figure area to mxn sections and plots the subsequent commands in section k
- `axis([XMIN XMAX YMIN YMAX])` assign the axis limit manually
- `xlim([XMIN XMAX])` assign the x-axis limit manually
- `ylim([YMIN YMAX])` assign the y-axis limit manually
- `legend(‘text’)` creates a legend for the current plot
Symbolic Math

• **syms** alerts MATLAB that you are using a symbolic variable, and that it does not have a specific value.

• **ezplot** plots symbolic functions

• **subs** substitutes a number or a symbol into a symbolic expression

• **simplify** simplifies a symbolic expression

• **int** calculates the symbolic integration

• **diff** calculates the symbolic differentiation
Symbolic Math Examples

- syms x y
- y = cos(x)
- ezplot(y,[0:0.1:2*pi])
- a = [0:0.1:2*pi];
- b = subs(y,x,a);
- plot(a,b)
- y = (x^2+5*x+6)/(x+2)
- z = simplify (y)
- diff(z)
- int(z)
Programming

• Expressions
• Flow Control blocks
  – Conditional
  – Iterations
• Scripts
• Functions
## Programming Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>~=</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Logical Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &amp;&amp; B</td>
<td>Logical and</td>
</tr>
<tr>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>
Programming Flow Control

if expression1
    statements1
elseif
    expression2
    statements2
else
    statements3
end

for variable = initval:endval
    statement
    ...
    statement
end

while expression
    statements
end
Programming Examples

```
a = input('input a? ');  
b = input('input b? '); 
if a == b,  
    fprintf('a is equal to b
');  
elseif a > 0 && b > 0  
    fprintf('both positive
');  
else  
    fprintf('other case
');  
end

M = rand(4,4);  
i = 1; j = 1; suma = 0;  
while i <= 4  
    while j <= 4  
        suma = suma + M(i,j);  
        j = j+1;  
    end  
    i = i+1;  
end  
fprintf('sum = %d
',suma);
```

```
M = rand(10,10); suma = 0;  
for i = {2,5:8}  
    for j = {1:5,8:9}  
        suma = suma + M(i,j);  
    end  
end  
fprintf('sum = %d
',suma);
```

```
M = rand(4,4); suma = 0;  
for i = 1:4  
    for j = 1:4  
        suma = suma + M(i,j);  
    end  
end  
fprintf('suma = %f
',suma);
```

```
M = rand(10,10); suma = 0;  
for i = 1:4  
    for j = 1:4  
        suma = suma + M(i,j);  
    end  
end  
fprintf('sum = %d
',suma);
```
Creating an m-file

• scripts/functions are usually referred to as m-file in MATLAB because they are saved with *.m extension
• The name of an M-file must begin with an alphabetic character
• scripts are simply pieces of code that are organized inside a file (i.e. they could be run line-by-line inside the MATLAB workspace) so they can add to workspace variables
• functions cannot be run separately but they have to be called inside other scripts/functions and their variables are local
Writing your first m-file (script)

% This script creates a NxN multiplication table
clc
clear all
N = input('Please enter size : ');
if (N > 10)
    disp('N must be less than 10');
else
    for i=1:N
        for j=1:N
            z = i*j;
            fprintf('%5d',z);
        end
        fprintf('
');
    end
end
% This function adds the divisors of N
% e.g. N = 10 -> 1+2+5 = 8
function result = divisor_sum(N)
sum = 0;
for i=1:N-1
    if mod(N,i)== 0
        sum=sum+i;
    end
end
result = sum;
References