Informatics 1 CG
Introduction to Matlab

Carina Silberer

Lab 1 – Week 3

January 22, 2014
Matlab

- a high-level language
- interactive environment (you can program on the fly)
- for numerical computation, visualization, programming
- command-line interface and GUI
- commercially developed by MathWorks
- open source variant: GNU octave
  highly compatible with Matlab
Getting started

Starting Matlab

From Dice terminal - opens GUI

[<name>]<studentID>: matlab

or for command-line interface

[<name>]<studentID>: matlab -nodisplay

Exiting Matlab

>> exit
# Arithmetic Operations and Functions

## Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 + 10</td>
<td>Addition</td>
</tr>
<tr>
<td>8 − 10</td>
<td>Subtraction</td>
</tr>
<tr>
<td>5 * 14</td>
<td>Multiplication</td>
</tr>
<tr>
<td>7/8</td>
<td>Division</td>
</tr>
<tr>
<td>$7^3$</td>
<td>Exponentiation</td>
</tr>
<tr>
<td>3.8 * 10^3</td>
<td>Exponentiation</td>
</tr>
<tr>
<td>3.8 * 10^{-3}</td>
<td>Exponentiation</td>
</tr>
</tbody>
</table>

## Built-in in matlab

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;&gt; 8+10</td>
<td>Addition</td>
</tr>
<tr>
<td>&gt;&gt; 8−10</td>
<td>Subtraction</td>
</tr>
<tr>
<td>&gt;&gt; 5*14</td>
<td>Multiplication</td>
</tr>
<tr>
<td>&gt;&gt; 7/8</td>
<td>Division</td>
</tr>
<tr>
<td>&gt;&gt; 7^3</td>
<td>Exponentiation</td>
</tr>
<tr>
<td>&gt;&gt; 3.8e3</td>
<td>Exponentiation</td>
</tr>
<tr>
<td>&gt;&gt; 3.8e−3</td>
<td>Exponentiation</td>
</tr>
</tbody>
</table>

## Some built-in functions in matlab

- >> sin(x)
- >> exp(x)

To list all built-in elementary functions

>> help elfun
Data Types (1)

Numeric Types – Integer and floating-point numbers

```
>> 344
>> 3.44
```

Booleans (logicals): 0 or 1 (false or true)

Characters and Strings

```
>> 'this is a string'
```

Matrices and Vectors

Use variables to store values.

```
>> a = 300/15-15
>> b = a*2.5
>> s = 'Hello, I am another string.'
```
Logical Operators

- not
- or
- and

- equal to
- not equal to
- less than
- greater than or equal to
**IF Statements**

```
>> a = 3;
>> if a>2
    disp('Decreasing a by 1.'
    a = a-1;
end
```

```
>> if a>2
    disp('Decreasing a by 1.'
    a = a-1;
else
    disp('a is smaller than or equal to 2.'
end
```
Loops

FOR-loop

for INDEX=START:END
    Execute this block for a fixed number of times.
end

>> a = 0;
>> for i=1:10
    disp('Increasing a by 1.')
    a = a+1;
end

WHILE-loop

while CONDITION
    Execute this block as long as condition is met.
end

>> a = 0;
>> while a<10
    disp('Increasing a by 1.')
    a = a+1;
end

Use break statement to leave loop mid-way;
Useful Commands

Use ; to suppress answer on display.

```matlab
>> a = 300/15;
```

Use disp() to display content.

```matlab
>> disp(a)
20
```

Ask for value of variable.

```matlab
>> a
```

Use clear to remove variable from memory.

```matlab
>> clear a
```

Clear complete memory.

```matlab
>> clear all
```

Use % for comments.

```matlab
>> c = (10-13)*5 % should be -15
```
Data Types (2) – Matrices

Vectors

\[ V_1 = \begin{pmatrix} 1 & 2.1 & -3 \end{pmatrix} \]
\[ V_2 = \begin{pmatrix} 1 \\ 2.1 \\ -3 \end{pmatrix} \]

\[
\text{>> } V_1 = \begin{bmatrix} 1, 2.1, -3 \end{bmatrix} \\
\text{>> } V_2 = \begin{bmatrix} 1; 2.1; -3 \end{bmatrix}
\]

Vectors are one-dimensional matrices.

Matrices

\[
M = \begin{pmatrix}
1 & 2 & 3 & 4 \\
2 & 3 & 1 & 1 \\
3 & 1 & 2 & 2
\end{pmatrix}
\]

\[
\text{>> } M = \begin{bmatrix} 1, 2, 3, 4; 2, 3, 1, 1; 3, 1, 2, 2 \end{bmatrix} \\
\text{>> } M = \begin{bmatrix} 1 & 2 & 3 & 4; 2 & 3 & 1 & 1; 3 & 1 & 2 & 2 \end{bmatrix}
\]

Commas separating column elements can be omitted.
Accessing Matrix Elements (1)

\[ M = \begin{bmatrix} 1 & 8 & 6; 2 & 4 & 7; 5 & 9 & 3 \end{bmatrix}; \]

Use indexing to access matrices, where index starts from 1.

\[ M_{i,j}, \text{i.e. element in row } i, \text{ column } j \]

\[ >> M(2,3) \% \text{ row 2, column 3} \]
\[ 7 \]

\[ >> M(3) \% \text{ row 3, column 1} \]
\[ 5 \]

Use end keyword to indicate last index.

\[ >> M(3,\text{end}) \% \text{ row 3, last column} \]
\[ 3 \]

\[ >> M(\text{end-1,2}) \% \text{ row 2, column 2} \]
\[ 4 \]
Accessing Matrix Elements (2)

Use colon : to select columns or rows.

\[ M = \begin{bmatrix} 1 & 8 & 6; & 2 & 4 & 7; & 5 & 9 & 3 \end{bmatrix}; \]

- \( M_{i,:} \): complete row \( i \)
- \( M_{:,j} \): complete column \( j \)

\[ \begin{align*}
\gg M(2,:) & \quad \text{row} \, 2 \\
\text{ans} & = \\
& \begin{bmatrix} 2 & 4 & 7 \end{bmatrix} \\
\gg M(:,3) & \quad \text{column} \, 3 \\
\text{ans} & = \\
& \begin{bmatrix} 6 \\ 7 \\ 3 \end{bmatrix}
\end{align*} \]

\[ \begin{align*}
\gg M(2:3,:) & \\
\text{ans} & = \\
& \begin{bmatrix} 2 & 4 & 7 \\ 5 & 9 & 3 \end{bmatrix}
\end{align*} \]

\[ \begin{align*}
\gg S & = M(2:end,1:2) \\
S & = \\
& \begin{bmatrix} 2 & 4 \\ 5 & 9 \end{bmatrix}
\end{align*} \]
Side note on Colon

With colon a new vector can be created ranging from a:c

\[
\begin{align*}
\text{>> } V &= 10:15 \text{ } \% \text{ } a = 10, \text{ } c = 15 \\
V &= \\
&= 10 \quad 11 \quad 12 \quad 13 \quad 14 \quad 15
\end{align*}
\]

With a:b:c an increment is specified.

\[
\begin{align*}
\text{>> } W &= 10:1.5:15 \text{ } \% \text{ } b = 1.5 \\
W &= \\
&= 10.0000 \quad 11.5000 \quad 13.0000 \quad 14.5000
\end{align*}
\]
Modifying Matrices (1)

\[
\begin{bmatrix}
1 & 8 & 6 \\
2 & 4 & 7 \\
5 & 9 & 3
\end{bmatrix}
\]

Replace items directly.

\[
M_{row,column} = new\_value \\
M_{s:e,c:d} = new\_values
\]

Add or remove a column or row.

\[
M(:,end+1) = [4; 5; 4] \\
M(2,:) = []
\]

\[
\begin{bmatrix}
1 & 8 & 9 & 4 \\
5 & 9 & 3 & 4
\end{bmatrix}
\]
Modifying Matrices (2)

Concatenate matrices (or vectors).

\[
\begin{bmatrix}
1 & 8; 2 & 7; \v1 = [5; 9]; \v2 = [13 0 0]; \v3 = [0 1 2];
\end{bmatrix}
\]

\[
\begin{bmatrix}
\end{bmatrix}
\text{Concatenate horizontally}
\]

\[
\begin{bmatrix}
\end{bmatrix}
\text{Concatenate vertically}
\]
Mathematical Operations with Matrices (1)

>> A = [4 5; 6 7]; B = [3 8; 9 1]; C = [2,3,4;5,5,1];

Sum and difference works just as with scalars.

A + B
A - B

>> disp(A - B)

1    -3
-3     6

Beware that matrices need to be of same dimensions.

>> disp(A - C)

??? Error using ==> plus
Matrix dimensions must agree.

>> disp(size(A))

2    2

>> disp(size(C))

2    3
Mathematical Operations with Matrices (2)

>> A = [4 5; 6 7]; B = [3 8; 9 1]; C = [2,3,4;5,5,1];

Product

A · B  Matrix product
A · b  Product with scalar
A^b   Matrix power

>> disp(A*B)
   57   37
   81   55

>> disp(A*C)
   33   37   21
   47   53   31

>> P = B^3;
>> S = B*1.3;

Element-wise operations

(A_{ij} × B_{ij})_{i=1,...,n;j=1,...,m}
(A_{ij}^b)_{i=1,...,n;j=1,...,m}
(A_{ij} ÷ B_{ij})_{i=1,...,n;j=1,...,m}

>> disp(A.*B)
   12   40
   54   7

>> disp(A.*C)
??? Error using ==> times
Matrix dimensions must agree.

>> P2 = A.^3;
>> D = A./B;
Useful Matrix Functions (1)

```matlab
>> M = [2,3,4;5,5,1];
```

Size of M.

```matlab
>> size(M);
```

Transpose of M.

```matlab
>> disp(M') % == transpose(M);
    2  5
    3  5
    4  1
```

Sum of rows.

```
\[ \sum_{i=1}^{n} M_{i,:} \]
```

Sum of columns.

```
\[ \sum_{j=1}^{m} M_{.,j} \]
```

```matlab
>> sum(M); % == sum(M,1);
    7  8  5
```

```matlab
>> sum(M'); % == sum(M,2);
    9
    11
```
Useful Matrix Functions (2)

\[
\begin{bmatrix}
2,3,4;5,5,1;8,0,1 \end{bmatrix}; \quad [1;44;3]; \quad [9;8;7];
\]

Diagonal of \( M \).

\[
\text{disp(diag}(M))
\]

\[
\begin{bmatrix}
2 \\
5 \\
1
\end{bmatrix}
\]

Upper triangular part of \( M \).

\[
\text{disp(triu}(M))
\]

\[
\begin{bmatrix}
2 & 3 & 4 \\
0 & 5 & 1 \\
0 & 0 & 1
\end{bmatrix}
\]

Lower triangular part of \( M \).

\[
\text{tril}(M);
\]

\[
\sum_{i=1}^{n} V_1_i V_2_i \quad \text{Vector dot product.}
\]

\[
\text{disp(dot}(V_1,V_2))
\]

\[
382
\]
Useful Matrix Functions (3)

>> M = [2,3,4;5,5,1;8,0,1];

Replicate and tile matrices: \texttt{repmat}(M, [a,b])

>> \texttt{disp(repmat(M, [1,3]))} \% 1-by-3 tiling
   2  3  4  2  3  4  2  3  4
   5  5  1  5  5  1  5  5  1
   8  0  1  8  0  1  8  0  1

>> \texttt{disp(repmat(M, [2,1]))} \% 2-by-1 tiling
   2  3  4
   5  5  1
   8  0  1
   2  3  4
   5  5  1
   8  0  1
Useful Matrix Functions (4)

Create specialized n-by-m matrices with $func(n, m)$

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

```matlab
>> ones(3,4);
```

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```matlab
>> eye(3); % n=m=3
```

$$M = (0 \ 0 \ 0 \ 0)$$

```matlab
>> zeros(1,4);
```

Create pseudorandom matrix.

```matlab
>> disp(rand(3,4))
0.3922 0.7060 0.0462 0.6555 0.0318 0.0971 0.1712 0.2769 0.8235
```
Plot 2-D lines with `plot(X,Y)`
X specifies locations along x-axis, and Y gives data values to plot.

```matlab
>> X = 1:.1:100;
>> Y = cos(X);
>> figure
>> plot(X,Y)
```
Set axis limits with \texttt{axis([x1 \ x2 \ y1 \ y2])}
Add title and labels with `title(str), xlabel(str), ylabel(str)`

```plaintext
>> X = 1:.1:100; Y = cos(X); figure; plot(X,Y)
>> axis([0 10*pi -1.5 1.5]);
>> title('Cosinus Function', 'fontsize', 10);
>> xlabel('angle');
>> ylabel('cos(x)');
```
Plot several 2-D lines with `plot(X1,Y1, ..., Xn,Yn)`

```matlab
>> X = 1:.1:100; Y1 = cos(X); Y2 = sin(X); figure;
>> plot(X,Y1,X,Y2)
```

Specify line style, marker, and colour with
`plot(X1,Y1,'LSpec', ..., Xn,Yn,'LSpec')`

```matlab
>> plot(X,Y1,'--r',X,Y2,'x') % red, dashed line; crosses
```
Add legend with `legend('Legend1', ..., 'Legendn')`

```matlab
>> legend('cos(x)','sin(x)')
```
Plot discrete data as a bar chart with `bar(Y)` or `bar(X,Y)`

The table lists the countries with the highest annual per capita consumption of tea in kg (2009):

<table>
<thead>
<tr>
<th>country</th>
<th>Turkey</th>
<th>Morocco</th>
<th>Ireland</th>
<th>Mauritania</th>
<th>UK</th>
</tr>
</thead>
<tbody>
<tr>
<td>kg</td>
<td>6.87</td>
<td>4.34</td>
<td>3.22</td>
<td>3.22</td>
<td>2.74</td>
</tr>
</tbody>
</table>

[a]wikipedia.org/wiki/List_of_countries_by_tea_consumption_per_capita

```matlab
>> Y = [6.87, 4.34, 3.22, 3.22, 2.74];
>> X = {'Turkey','Morocco','Ireland','Mauritania','UK'};
>> bar(Y)
>> ylabel('kg')
>> title('Tea consumption per capita')
>> axis([0 length(X)+1 0 8])
>> set(gca, 'XTickLabel', X) % set country labels
```
Plotting Discrete Data

Tea consumption per capita

Turkey: 7 kg
Morocco: 4 kg
Ireland: 3 kg
Mauritania: 3 kg
UK: 2 kg
Define function name, inputs and outputs with

```
function [y1,...,yN] = func_name(x1,...,xM)
```

- Function codes are saved in files with extension `.m`
- Multiple functions can be declared in one file.
- The file name should match the name of the first function in the file.

**Define function in file my_sum.m**

```
function [sum] = my_sum(x)
    sum = 0;
    for i=1:length(x)
        sum = sum+x(i);
    end
```

**Call function from command line**

```
>> s = my_sum([1,2,3,4.4]);
>> s
s =
    10.4000
```
The previously defined function `my_sum` only works for row vectors (1-by-m matrices):

```matlab
>> s = my_sum([1,2;4,5])
s =
   5
```

Modify the code so that it can compute the sum of the rows of n-by-m matrices:

```matlab
>> s = my_sum([1,2;4,5])
s =
   5   7
```