Course unit title:
Basics of Information Systems
Course unit code:
NIRIA1SEND

Matlab basics I.

Dr. habil. Levente Kovács
associate professor

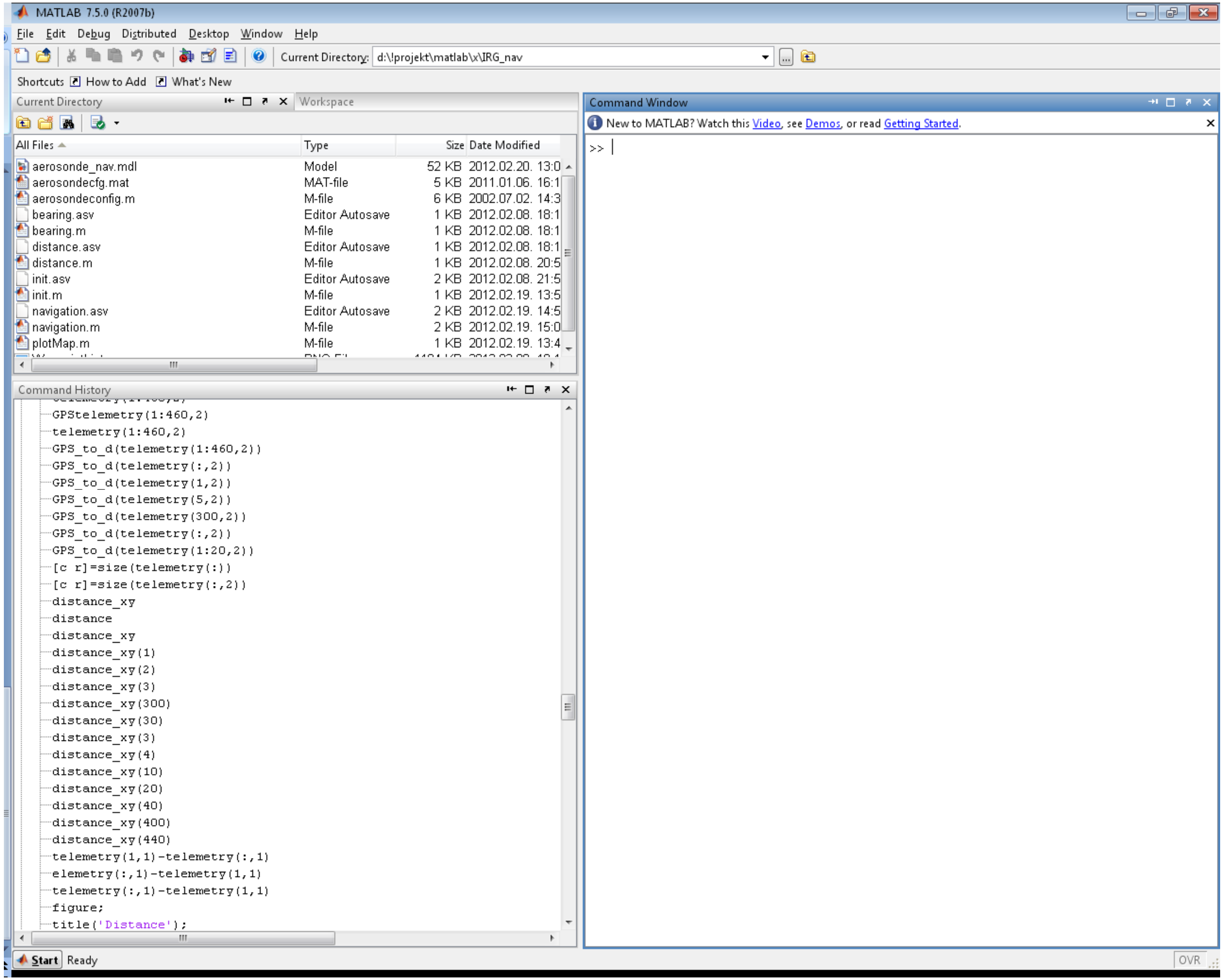
Obuda University,
John von Neumann Faculty of Informatics,
Department of Information Systems,
Physiological Controls Group

09.10.2013.

Literature

- S. Gisbert: MATLAB, Typotex Kft, Budapest, 2005, ISBN 963 9548 49 9
  - *(Google Books)*
- *MATLAB help*

Command Window



First steps – the very basics

- Commands -> Command Window
- More commands -> .m file
- Importance of ;
  - If not used, execution and result is displayed
  - If used, execution and result is NOT displayed
- Comment %
- Indexing starts from 1
- Help help ☺

Command Window

- Create a variable named *a*

```
a = 1
a =
     1
```

MATLAB immediately adds variable a to the workspace and displays the result in the Command Window.

- When you do not specify an output variable, MATLAB uses the variable `ans`, short for *answer*, to store the results of your calculation

```
sin(a)
ans =
          0.8415
```

The value of `ans` changes with every command that returns an output value that is not assigned to a variable.

If you end a statement with a semicolon, MATLAB performs the computation, but suppresses the display of output in the Command Window.

```
b = 2;
```

Command Window

```
>> a=10
a =
   10

>> A=1
A =
    1

>> A+a
ans =
   11
```

```
>> a= [0,1,2]
a =
     0     1     2

>> a= [0,1,2]'
a =
     0
     1
     2
```

Command Window

```
>> M=[1,2,3;4,5,6;7,8,9]

M =

    1    2    3
    4    5    6
    7    8    9
```

Command Window

```
>> a= [0:0.1:1]’
a =
         0
    0.1000
    0.2000
    0.3000
    0.4000
    0.5000
    0.6000
    0.7000
    0.8000
    0.9000
    1.0000
```

Command Window

```
>> length(a)
ans =
    11

>> b=-1
b =
    -1

>> abs(b)
ans =
     1
```

Command Window

```
>> ones(3)
ans =
    1    1    1
    1    1    1
    1    1    1

>> zeros(3)
ans =
    0    0    0
    0    0    0
    0    0    0
```

Command Window

```
>> a = rand(3)

a =

    0.9649    0.9572    0.1419
    0.1576    0.4854    0.4218
    0.9706    0.8003    0.9157


>> a=a*10

a =

    9.6489    9.5717    1.4189
    1.5761    4.8538    4.2176
    9.7059    8.0028    9.1574
```

Command Window

```
>> a = rand(3,1)
a =
    0.0357
    0.8491
    0.9340

>> a=a*10
a =
    0.3571
    8.4913
    9.3399
```

```
>> min(a)
ans =
    0.3571

>> max(a)
ans =
    9.3399

>> size(a)
ans =
     3     3
```

.m files

What is an m-file?

• an m-file, or script file, is a simple text file where you can place MATLAB commands

• all m-file names must end with the extension '`.m`' (e.g. `test.m`)

Why use m-files?

• for simple problems, entering your requests at the MATLAB prompt is fast and efficient

• for long, complex problems m-files are very helpful and almost necessary

How to run the m-file?

• after the m-file is saved with the name `filename.m` in the current MATLAB folder or directory, you can execute the commands in the m-file by simply typing filename at the MATLAB command window prompt

Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
'	Complex conjugate transpose
*	Matrix multiplication
/	Matrix right division
\	Matrix left division
^	Matrix power

Operator	Description
.*	Multiplication
./	Right division
.\	Left division
.^	Power
.'	Transpose

Relational Operators

Operator	Description
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to

Logical Operators

<table>
<tr><th>Operator</th><th>Description</th><th>Example</th></tr>
<tr><td>&</td><td>Returns 1 for every element location that is true (nonzero) in both arrays, and 0 for all other elements.</td><td>A & B = 01001</td></tr>
<tr><td>|</td><td>Returns 1 for every element location that is true (nonzero) in either one or the other, or both arrays, and 0 for all other elements.</td><td>A | B = 11101</td></tr>
<tr><td>~</td><td>Complements each element of the input array, A.</td><td>~A = 10010</td></tr>
<tr><td>xor</td><td>Returns 1 for every element location that is true (nonzero) in only one array, and 0 for all other elements.</td><td>xor(A,B) = 10100</td></tr>
</table>

- The examples shown in the following table use vector inputs A and B, where `A = [0 1 1 0 1]; B = [1 1 0 0 1];`
- For operators and functions that take two array operands, (&, |, and xor), both arrays must have equal dimensions, with each dimension being the same size.

who, whos, clear, clc

`who`: lists in alphabetical order all variables in the currently active workspace

```
Your variables are:
A    B    a    ans  b
```

`whos`: display information about all variables in the currently active workspace

```
Name       Size            Bytes  Class     Attributes
  A         1x1            8   double
  B         1x1            8   double
  a         1x1            8   double
  ans       1x1            1   logical
  b         1x1            8   double
```

`clear`: removes all variables from the current workspace, releasing them from system memory

`clc`: clears all input and output from the Command Window display, giving you a "clean screen.”

disp

`disp(X)`: displays the contents of X without printing the variable name. disp does not display empty variables

```
disp('text')
text

s= 'text'
s =
   text

disp(s)
text
```

Calculations on Vectors and Matrices

- Whereas some MATLAB functions support only vector inputs, others accept matrices.
- When your data is a vector, the result is the same whether the vector has a rowwise or columnwise orientation.
- When your data is a matrix where each row contains a data set,
    - ✓ you must transpose the matrix before proceeding with the data-analysis tasks to make the data sets have a columnwise orientation
    - ✓ eg. to transpose a real matrix A, use the syntax `A'`.

Creating Vectors

- Enter each element of the vector (separated by a space) between brackets, and set it equal to a variable.

```
a = [1 2 3 4 5 6 9 8 7]
a =
   1 2 3 4 5 6 9 8 7
```

- Create a vector with elements between 0 and 20 evenly spaced in increments of two (this method is frequently used to create a time vector):

```
t = 0:2:20
t =
    0 2 4 6 8 10 12 14 16 18 20
```

Manipulating Vectors

- Add 2 to each of the elements in the vector **a**

```
b = a + 2
b =
    3 4 5 6 7 8 11 10 9
```

- Add two vectors together (if the two vectors are the same length)

```
c = a + b
c =
    4 6 8 10 12 14 20 18 16
```

- Find the transpose of a vector using the apostrophe key

```
d = c'
```

Creating Matrices

- The same as entering a vector, except each row of elements is separated by a semicolon (`;`) or a return:

```
B = [1 2 3 4; 5 6 7 8; 9 10 11 12]
B = [ 1  2  3  4
      5  6  7  8
      9 10 11 12 ]
```

```
B = 1  2  3  4
    5  6  7  8
    9 10 11 12
```

Manipulating Matrices

- Find the transpose of a matrix using the apostrophe key

```
C = B'

C = 1 5 9
    2 6 10
    3 7 11
    4 8 12
```

Manipulating Matrices

- Multiply the two matrices *B* and *C* together. Remember that order matters when multiplying matrices!

```
D = B * C
D = C * B
```

```
D = 30  70  110
    70  174 278
    110 278 446

D = 107 122 137 152
    122 140 158 176
    137 158 179 200
    152 176 200 224
```

Manipulating Matrices

- Multiply the corresponding elements of two matrices using the `.*` operator (the matrices must be the same size to do this)

```
E = [1 2; 3 4]
F = [2 3; 4 5]
G = E .* F
```

```
E = 1 2
    3 4
F = 2 3
    4 5
G = 2  6
    12 20
```

Physiological Controls Group

Manipulating Matrices

- Multiply the matrix by itself (the matrix must be a square matrix)

```
E^3
ans =
      37   54
      81  118
```

- To cube each element in the matrix, just use the element-by-element cubing

```
E.^3
ans =
      1   8
      27  64
```

Manipulating Matrices

- Find the inverse of a matrix

```
X = inv(E)
X =
    -2.0000 1.0000
     1.5000 -0.5000
```

- Find the eigenvalues of a matrix

```
eig(E)
ans =
     -0.3723
      5.3723
```

zeros, ones, eye, rand

zeros(n): returns an n-by-n matrix of zeros (**zeros** returns the scalar 0)

```
zeros(2)
ans =

     0     0
     0     0
```

ones(n): returns an n-by-n matrix of ones (**ones** returns the scalar 1)

```
ones(2)
ans =

     1     1
     1     1
```

eye(n): returns an n-by-n identity matrix with ones on the main diagonal and zeros elsewhere (**eye** returns the scalar 1)

```
eye(2)
ans =
     1     0
     0     1
```

rand(n): returns an n-by-n matrix containing pseudorandom values drawn from the standard uniform distribution on the open interval (0,1).

```
rand(2)
ans =

    0.8147    0.1270
    0.9058    0.9134
```

Elements of a matrix

`a (:)` each element - 1D
`a (:, :)` each element - 2D
`a = (1,1)` 1,1 element
`a = (:, 1)` 1. column
`a = (1, :)` 1. row
`d = size(X)` returns the sizes of each dimension of array **X** in a vector *d*
`[m,n] = size(X)` returns the size of matrix **X** in separate variables *m* and *n*
`length(vector)` returns the length of the vector
`length(array)` finds the number of elements along the largest dimension of an array

if, elseif, else, end

```
if expression
   statements
elseif expression
   statements
else
   statements
end
```

- `if expression, statements, end` evaluates an expression, and executes a group of statements when the expression is true.
- `elseif` and `else` are optional, and execute statements only when previous expressions in the `if` block are false; an `if` block can include multiple `elseif` statements
- an evaluated expression is true when the result is nonempty and contains all nonzero elements (logical or real numeric); otherwise, the expression is false.
- expressions can include relational operators (such as `<` or `==`) and logical operators (such as `&&`, `||`, or `~`)

for, while

```
for index = values
    program statements
           :
end
```

• repeatedly executes one or more MATLAB statements in a loop

```
for s = 1.0: -0.1: 0.0
    disp(s)
end
```

• step by increments of -0.1, and display the step values

```
while expression
      statements
end
```

• repeatedly executes one or more MATLAB program *statements* in a loop as long as an expression remains true

Importing Data

MATLAB Import Wizard

✓you can import the following types of data sources:

- Text files, such as `.txt` and `.dat`
- MAT-files
- Spreadsheet files, such as `.xls`
- Graphics files, such as `.gif` and `.jpg`
- Audio and video files, such as `.avi` and `.wav`

Loading Data

```
load count.dat
```

- import data into MATLAB using the `load` function
- loading this data creates a 24-by-3 matrix called count in the MATLAB workspace

```
[n,p] = size(count)
n =
      24
p =
      3
```

- get the size of the data matrix
- *n* represents the number of rows, and *p* represents the number of columns

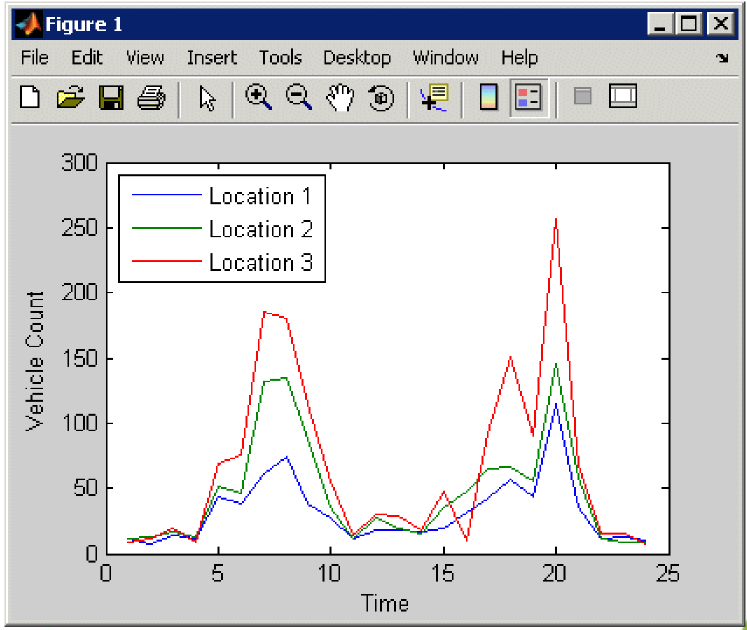
Plotting Data

```
t = 1:n;
```

- create a time vector, *t*, containing integers from 1 to *n*

```
plot(t,count),
legend('Location 1','Location 2','Location 3',2)
xlabel('Time'), ylabel('Vehicle Count')
```

- plot the data as a function of time,
- and annotate the plot



Exercise 1

Calculating π with Leibniz series

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$$

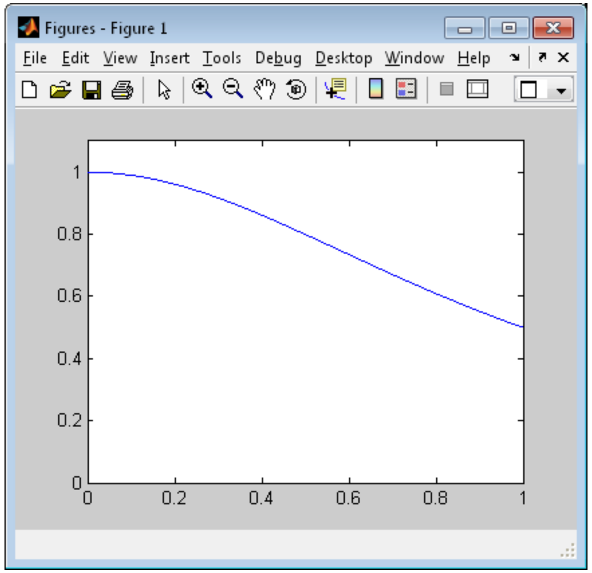
```
format long
iterPi = 10000000; % iteration
erLeibniz = 1; %1. element of the series
for i = 3:4:iterPi
    erLeibniz = erLeibniz - (1.0 / i) + (1.0 / (i + 
2));
end

disp('Eredmény: ')
erLeibniz=erLeibniz*4

disp('Pi:')
pi
```

Exercise 2

Calculating π with $\frac{1}{1+x^2}$ area under the function

$$\int_0^1 \frac{1}{1+x^2}\, dx = \frac{\pi}{4} \approx 0.785398$$

```
x=[0:0.001:1];
y= 1./(1+x.^2);
plot(x,y)
xlim([0 1]);
ylim([0 1.1]);
```



Exercise 2

Calculating π with $\frac{1}{1+x^2}$ area under the function

```
myfun.m:
     function y = myfun(x)
     y= 1./(1+x.^2) *4;

>> quad(@myfun,0,1)

ans =

   3.14159268292457
```

Exercise 2

Calculating π with $\frac{1}{1+x^2}$ area under the function

```
format long
iterPi = 10000000; % iteration
x=0;
width =  1 / iterPi;
erFuggvenyTer = 0;

for  i = 0:iterPi
    x = (i + 0.5) * width;
    erFuggvenyTer = erFuggvenyTer + 4 / (1 + x * x);
end

disp('1/(1+n^2) alatti terület : ')
erFuggvenyTer*width

disp('Pi:')
pi
```

Course unit title:
Basics of
Information
Systems
Course unit code:
NIRIA1SEND

Thank you for your attention!

kovacs.levente@nik.uni-obuda.hu

09.10.2013.