Artificial Intelligence

Lab 1 manual

"Introduction to Matlab"

1 Basic Commands

Before you start working with with the assignments it is strongly suggested to set the path to the folder where you can save your work (Desktop or memory stick). The below mentioned commands will allow all of you to get familiar with Matlab and its Toolboxes.

1.1 Moving around

`pwd` - displays the current location
`dir` - lists files in the current folder
`cd` - folder change

Now you can move the location when you want to save files from the current lab session.

1.2 Getting help

In Matlab you can get help for all the commands by typing:

>> `help yourcommand`

Example: >> `help help`

Now you can get desired information about command 'help'.

1.3 Command `lookfor`

Command `lookfor` is useful in the situations when we don't remember the name of the command that we want to use. To illustrate the usage of this function we will analyze searching for the function name that is used to calculate the cosine of an angle along with its associated functions:

>> `lookfor cosine`

When the above command is typed in Matlab all the functions associated with cosine will be displayed. You can also notice that in addition all the associated with cosine functions are also return. See the following example:

`acos`	- Inverse cosine, result in radians.
`acosd`	- Inverse cosine, result in degrees.
`acosh`	- Inverse hyperbolic cosine.
`cos`	- Cosine of argument in radians.
`cosd`	- Cosine of argument in degrees.
`cosh`	- Hyperbolic cosine.
`slsincos`	- This is a private mask helper file for sine and cosine blocks in
`slsincoslut`	- This is a private mask helper file for sine and cosine blocks in
`fi_cordic_sincos_demo`	- Compute Sine and Cosine Using CORDIC Rotation Kernel
`fi_sin_cos_demo`	- Calculate Fixed-Point Sine and Cosine
`dct2`	- 2-D discrete cosine transform.
`dctmtx`	- Discrete cosine transform matrix.
`idct2`	- 2-D inverse discrete cosine transform.
`mech_car_raised_cosine`	- Helper function to create a raised cosine pulse at time 'T'
`chirp`	- Swept-frequency cosine generator.
`dct`	- Discrete cosine transform.
`firrcos`	- Raised Cosine FIR Filter design.
`idct`	- Inverse discrete cosine transform.
`tffunc`	- time and frequency domain versions of a cosine modulated Gaussian pulse.
`cosint`	- Cosine integral function.

1.4 Matrices in Matlab

Matrices (and vectors) are the basic data structure in all Matlab programs. In contrary to other programming languages doesn't require any loops to perform matrix operations. This is a very useful especially when working with pattern recognition and image processing. For better understanding of Matlab Matrix operations, please look at the following examples:

Type:

```
>> x = [1 2 3 4 5 4 3 2 1];
```

Creates a 1x9 vector represented by variable x and holds the above mentioned integers.

```
>> x
```

Displays values stored in x

```
>> who
```

Allows for displaying informations about all active matrices/vectors.

```
>> whos
```

Displays even more detailed informations.

```
>> y = [6; 7; 8; 9; 0; 9; 8; 7; 6]
```

Creates a 9x1 matrix.

```
>> y'
```

' is a transpose operations

```
>> z = [1 2 3; 4 5 6; 7 8 9; 0 1 2]
```

2 Operators

2.1 Operator ':'

Allows for a fast definitions of some vectors:

```
>> a = 0:10
>> b = -5:10
>> c = 0:2:10
>> d = 10:-2:-5
>> e = 0:0.01:4.2
>> f = -pi:0.01:pi
```

The above instructions are of the form: *starting value:increment:end value Default value for increment is 1. To create matrices containing zeros or onces only we will use zeros and ones accordingly. Function randn will be useful for creation of matrices with random values. Examples:*

```
>> g = zeros(2,4)
>> h = ones(5,3)
```

2.2 Arithmeetic operations

*Standard operations: + plus, - minus, / division, * multiplication, ^ power*
Try:

```
>> 3^2
>> y'
>> x + y'
>> x * y
>> 3 * x
```

*Operations /, * i ^ have can also are also defined for matrices: ./ , .* and .^ . These operations are performed element by element. Compare x*y and x.*y and check if the result is the same? For practice perform more comparisons:*

```
>> x^2
>> x.^2
>> x.*x
>> x*x
```

Now type in whos. As you can see, there are a few active variables. Please type in clear and then whos. You should notice that all the variables are gone.

Let's now create a matrix A:

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

and do the following:

```
>> sum(A) % Displays a column sum of A.
```

You should get:

```
>> 34  34  34  34
```

To calculate a sum for all the elements you have sum the above results:

```
>> sum(sum(A))
```

1. *Create a matrix J with all onces of a size 10x10 and a matrix with random numbers L of the same size and do the following:*
   (a) *A + L*
   (b) *A.L*
   (c) *L - A*
   (d) *L.A - is the result the same as in (b)?*
   (e) *L/A*
   (f) *A/L*
   (g) *A - L*
   (h) *sum of all elements of A and L*
2. *You were asked to analyze a financial situation of a certain organization. In files exp.txt and inc.txt you are given two vector with monthly income and outcome. You need to use this to calculate the following measures:*
   (a) *Revenue for each month*
   (b) *Revenue after tax (tax = 19%)*
   (c) *Revenue margin for each month - defined as a ration of gross revenue and income represented in %*
   (d) *Good months - where gross revenue > average yearly gross revenue*
   (e) *Bad months - where gross revenue < average yearly gross revenue*
   (f) *Best month in the year - where gross revenue is highest*
   (g) *Worst month in the year - where gross revenue is lowest*
   (h) *Results should be stored as a vector and the precision should be kept for 2 decimals point (ex. 10.01zł)*
   (i) *Remember that it is normal to get negative values for tax.*
   (j) *You should present the results at once.*