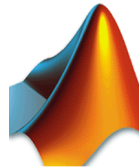


Matlab for Computer Science students

Lecture 3



Outline

- Format of the lecture:
 - Quick overview
 - Differences from C
- Why Matlab?
- Matrix notation
- Visualization
- Programming
- Miscellaneous

Why Matlab?

- Great tool for simulation and data analysis
- Concise matrix notation replaces loops
- Many easy to use "toolboxes"
- Easy visualization

Matlab environment

- Matlab is an interpreter by default
 - Although there is a possibility to compile
- Two modes
 - Command line
 - Scripts
- No need to declare variables
- Getting help:
 - `help function_name`

Matrix notation

- Every variable is a matrix: 2D double array
 - But of course 1D arrays and double can also be represented
- Outline
 - Defining matrices
 - Arithmetic
 - Matrix operation

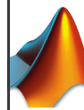
Defining variables

- The simplest matrix is 1x1 i.e. double number
- To create variable a equal to 1
 - `a = 1;`
 - This variable will be stored in a "workspace"
 - `a = 1`
 - Additionally its value will be printed on the screen:
 - `a=`
 - `1`
 - Expression without variable create default variable `ans`, for example typing "`2+2`" results in
 - `ans =`
 - `4`



Defining matrices

- To define horizontal vector
 - `a = [1, 2, 3];`
- To define vertical vector
 - `a = [1; 2; 3];`
 - In above two cases: To read out 2nd value: `a(2)`
 - Note: indexing from 1 not from 0
- To define a matrix 2 by 3:
 - `a = [1, 2, 3; 4, 5, 6]`
 - To read value in row 1, column 3: `value13=a(1,3)`
 - To read the first row: `row1=a(1,:)`



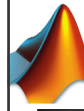
Matrices with sequences

- To define an arithmetic sequence
 - `myseq1 = [1, 2, 3, 4, 5];`
 - Simply type: `myseq1 = [1:5];`
- To define an arithmetic sequence
 - `myseq2 = [1.1, 1.2, 1.3, 1.4, 1.5];`
 - Simply type: `myseq2 = [1.1:0.1:1.5];`
- To concatenate the above two sequence
 - `myseq3 = [myseq1, myseq2]`
 - `myseq3 =`
`[1, 2, 3, 4, 5, 1.1, 1.2, 1.3, 1.4, 1.5]`



Predefined matrices

- To create a 3 by 3 matrix filled with zeros:
 - `a = zeros (3);`
- To create a 2 by 4 matrix filled with zeros:
 - `a = zeros (2, 4);`
- Other predefined matrix:
 - `ones` – filled with 1
 - `eye` – identity matrix (square only)
 - `rand` – filled with random numbers from uniform distribution between 0 and 1
 - `randn` – filled with random numbers from normal distribution with mean 0 and std 1



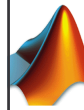
Matrix arithmetic

- Consider: `a = rand(2)`, `b = rand(2)`
- Arithmetic operation:
 - `a+b`, `a-b` – sum or difference between matrices
 - `a*b` – matrix multiplication
 - `a.*b` – element by element multiplication
 - `a./b` – element by element division
 - `a^2` – matrix multiplication `a*a`
 - `a.^2` – element by element squaring
 - `sin(a)` – 2 by 2 matrix containing sins of elements of matrix `a`, etc.



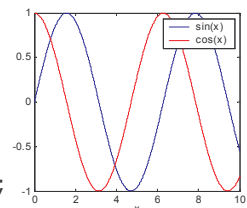
Matrix operations

- To get transpose of matrix `a`, type `a'`
- Consider vector `a`
 - `sum(a)` – sum of all elements in `a`
 - `mean(a)` – average value of elements in `a`
 - `std(a)` – standard deviation of elements in `a`
- For matrices the above functions create a horizontal vector containing sums/means/stds of columns of the matrix



Visualization: 2D -plots

- To plot `sin(x)`, and `cos(x)`, for `x ∈ (0,10)`
 - `x=[0:0.01:10];`
 - `y=sin(x);`
 - `plot(x,y);`
 - `z=cos(x);`
 - `hold on`
 - `plot(x,z,'r');`
 - `legend('sin(x)', 'cos(x)');`
 - `xlabel('x');`
- Also try editing plots in figure window

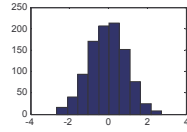




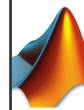
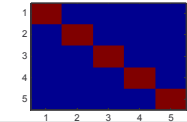
Visualization

- There is plenty of other really cool ways of visualization, my other favourites are:

- `hist(randn(1,1000));`



- `imagesc(eye(5));`



Scripts

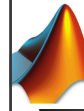
- To start editor type: `edit`
 - `function y = factorial (x)`
 - `% function y = factorial (x)`
 - `if x == 1 %really inefficient`
 - `y = 1;`
 - `else`
 - `y = x * factorial (x-1);`
 - `end`
- Should be saved with the same name:
 - `factorial.m`



Loops

- `i = 1;`
- `while i < 100`
- `i = i*2;`
- `end`

- `for i = [2:2:10]`
- `i`
- `end`



Miscellaneous

- Checking variables in workspace: `who, whos`
- Removing variables from workspace: `clear`
- Saving workspace: `save file_name`
- Loading workspace: `load file_name`
- Printing: `fprintf ('2+2=%d\n', 4);`
- Debugging: very easy debugger
 - See icons on the editor