GENERAL INFORMATION & MATLAB INTRODUCTION

Risk analysis lab
2017

Ceffer Attila
PhD student
BUTE Department Of Networked Systems and Services
ceffer@hit.bme.hu
General informations

- Lectures: Monday 13:15 - 14:45 (IL 108)
  - Prof. Janos LEVENDOVSZKY, levendov@hit.bme.hu
- Lab: Monday 15:15 - 16:45 (IL 108)
  - Attila CEFFER, ceffer@hit.bme.hu

- Final Grade = \( \frac{(\text{MidTermTest} + \text{LabAverage} + \text{Exam})}{3} \)
- LabAverage = Sum of lab points / Number of labs
- Number of labs = approx. 12

- The solutions of tasks must be uploaded using the Dropbox uploader form on the website.
- Participation is obligatory – max number of absence is 3.
<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017.09.04</td>
<td>MatLab introduction – not graded</td>
</tr>
<tr>
<td>2017.09.11</td>
<td>MatLab introduction 2</td>
</tr>
<tr>
<td>2017.09.18</td>
<td>MatLab introduction 3</td>
</tr>
<tr>
<td>2017.09.25</td>
<td>Calculating risk</td>
</tr>
<tr>
<td>2017.10.02</td>
<td>Chernoff bounds</td>
</tr>
<tr>
<td>2017.10.09</td>
<td>Chernoff bounds 2</td>
</tr>
<tr>
<td>2017.10.16</td>
<td>Chernoff bounds 3</td>
</tr>
<tr>
<td>2017.10.30</td>
<td>Generalized risk bandwidth</td>
</tr>
<tr>
<td>2017.11.06</td>
<td>Portfolio risk</td>
</tr>
<tr>
<td>2017.11.13</td>
<td>Trading with options</td>
</tr>
<tr>
<td>2017.11.20</td>
<td>Mean reverting portfolios</td>
</tr>
<tr>
<td>2017.11.27</td>
<td>Mean reverting portfolios 2</td>
</tr>
<tr>
<td>2017.12.04</td>
<td>Monte Carlo methods</td>
</tr>
</tbody>
</table>
MATLAB introduction - Agenda

- Introduction, architecture
- Data types
- Basic arithmetic operations and functions
- Random number generation
- Plotting, visualization
- Programming fundamentals
- Work with external data files
Introduction

- Cleve Moler (University of New Mexico) started developing in the late '70s
  - Recent versions: new release in every half year (R201x[a|b])

- Goal: to **perform numerical calculations**, mostly in engineering and research applications
  - Besides being a tool, MATLAB is also a programming language and a software platform
Architecture

Platform

- Built-in functions
- Toolboxes (e.g. Statistics and Machine Learning, Parallel Computing, Financial, Global Optimization, …)
  - Both command line and GUI interfaces
- Data handling framework
  - File I/O, database connection

Advantages

- Easy to learn
- Rapid prototype developing
- No need to hassle with low level issues
- Fast computation, optimized to the latest hardwares (e.g. GPGPU support)
Window layout

MATLAB R2013b

New to MATLAB? Watch this Video, see Examples, or read Getting Started.

A = [1 2 3; 4 5 6; 7 8 9]

ans =
1
2
3
4
5
6
7
8
9

A = [1 2 3; 4 5 6; 7 8 9]

ans =
18

mean(A)
ans =
4
5
6
7
8
9

MATLAB Introduction © Ceffer Attila BME HIT
Data representation

- Dynamic typing
  - No need to declare types
  - Type is assigned based on the right hand operand

- MATrix LABoratory
  - Numerical data are represented as complex matrices
    - Scalar values (1x1), vectors (1xN)

- Structs
  - {field1, value1, field2, value2, …}

- Other types: e.g. strings, database connections
### Basic operations (syntax)

- **Defining variables**

<table>
<thead>
<tr>
<th>Type</th>
<th>MATLAB code</th>
<th>Math</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalar (1x1 matrix)</td>
<td><code>c = 5;</code></td>
<td>$c = 5$</td>
</tr>
<tr>
<td>Row vector (1xN matrix)</td>
<td><code>v = [3 7 9];</code></td>
<td>$v = [3 7 9]$</td>
</tr>
<tr>
<td>Column vector (Nx1 matrix)</td>
<td><code>w = [4; 6; 8];</code></td>
<td>$w = \begin{bmatrix} 4 \ 6 \ 8 \end{bmatrix}$</td>
</tr>
<tr>
<td>Matrix (NxM)</td>
<td><code>A = [1 2 3; 4 5 6; 7 8 9];</code></td>
<td>$A = \begin{bmatrix} 1 &amp; 2 &amp; 3 \ 4 &amp; 5 &amp; 6 \ 7 &amp; 8 &amp; 9 \end{bmatrix}$</td>
</tr>
</tbody>
</table>
Basic operations (syntax)

- **Indexing**

\[
A = [1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 9];
\]

- Starts from 1, not from 0
- Select the whole row/column with operator:
  - `end` keyword to index last row/column
- Select intervals:
  - `A(2:3, 1)`
Basic operations (syntax)

- Calling functions

```
[d, e] = function_name(a, b, c);
```

- Arbitrary number of arguments
- User defined functions

- Some built-in functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>min(x)</code></td>
<td>Smallest elements in array</td>
</tr>
<tr>
<td><code>max(x)</code></td>
<td>Largest elements in array</td>
</tr>
<tr>
<td><code>abs(x)</code></td>
<td>Absolute value</td>
</tr>
<tr>
<td><code>sqrt(x)</code></td>
<td>Square root</td>
</tr>
<tr>
<td><code>exp(x)</code></td>
<td>Exponential</td>
</tr>
<tr>
<td><code>floor(x)</code></td>
<td>Round toward neg. infinity</td>
</tr>
<tr>
<td><code>ceil(x)</code></td>
<td>Round toward pos. infinity</td>
</tr>
<tr>
<td><code>log(x)</code></td>
<td>Natural logarithm</td>
</tr>
<tr>
<td><code>sum(x)</code></td>
<td>Sum of array elements</td>
</tr>
<tr>
<td><code>mean(x)</code></td>
<td>Average value of array</td>
</tr>
<tr>
<td><code>std(x)</code></td>
<td>Standard deviation</td>
</tr>
<tr>
<td><code>eig(x)</code></td>
<td>Eigenvalues and eigenvectors</td>
</tr>
</tbody>
</table>
Basic operations - initializers

- Initializing matrices (faster than the dynamic case)
  - A = zeros(n, [m]);
  - A = ones(n, [m]);
  - A = eye(n, [m]);
- \([N, M] = \text{size}(A)\);
- \(+, -, *, /, ^\) operators
  - No C-style operators are provided like += or ++
  - \(A \backslash B = A^{-1}B\) (solves \(Ax = b\) linear equation system)
- Logical expressions
  - <, <=, ==, ~= (not ! =), >=, >, ~, &,
Basic operations

- Comments
  - `a = 1; % one line comment`
  - Multi-line comments: between `%{ and %}`
    (separate line)

- Semicolon (`;`) at the end of the command
  - If presented: perform the operation
  - If not presented: perform the operation and display the result on the console
Random number generation

- Uniformly distributed pseudorandom numbers in the open interval (0,1)
  - rand(n, [m])
  - Random numbers within a specific range [a, b]: \( r_{ab} = a + (b - a)r_u \)

- Normally distributed numbers (standard normal)
  - randn(n, [m])
  - Random numbers with specific mean (\( \mu \)) and variance (\( \sigma \)): \( r_{\mu\sigma} = \mu + r_n \sigma \)

- Other distributions
  - random(name, param1, [param2], n, [m])
    - 'Binomial', n, p
    - 'Exponential', mean
    - 'Geometric', p
    - 'Poisson', lambda
    - ..
Plotting

- New plot window: `figure;
  plot(x, y, [x2, y2, ...], [params]);`
  - `x → D, y → R`
  - Only dots: `plot(x, y, 's');`
  - Multiple curves in one plot: `hold on;
    plot(...); hold off;`
- `bar(x); hist(x);`

- Adding features
  - Grid: `grid;`
  - Titles: `title('abc'); xlabel('x');
    ylabel('y'); legend('a', 'b', 'c');`
  - Until opening a new window
- Exportable to static image files of various formats
Programming fundamentals

- Scripts
  - Series of commands
  - `.m` extension
  - Executed by its filename (without extension)
  - All variables will be global (workspace)
  - Must be in the current folder

- Functions
  - `.m` extension (must have the same name)
  - Invoked by its name with providing the input variables
  - All variables (including the input variables) will be local
  - Must be in the current folder

- Best practice: in order to provide maximal maintainability and reusability **put every functionality into separate functions** and use a script as a „main” function
Function declaration

```matlab
function [r1, r2, r3] = myfunction(a, b, c)
...
end
```

(multiple and named return values → no return statement needed, simply assign values to them)
**Conditionals**

```matlab
if expression1
    ...
elseif expression2
    ...
else
    ...
end
```

```matlab
switch
    case expression1
        ...
    case expression2
        ...
    otherwise
        ...
end
```
Loops

- Count-controlled loop

\[ v = 1:10; \]
\[ \text{for } i = v \]
\[ \quad \ldots \]
\[ \text{end} \]

- Condition-controlled loop

\[ \text{while } \text{expression} \]
\[ \quad \ldots \]
\[ \text{end} \]

- Terminate the loop: \texttt{break};
- Continue with the next cycle: \texttt{continue};
Debugging

- Live example
  - Write running sum task in RiskLab Tasks 1
  - Error
  - Breakpoints
  - Stop on error
Work with data files

- Binary format for storing variables: `.mat`
  - Global variables can be saved from / loaded to workspace
  - From command line
    `save('data.mat', 'a1', 'a2');`
    `load 'data.mat';`
  - Remove all variables: `clear;`
    (scripts should start with this to avoid „interference”)

- Loading data from `.csv` or `.xls` format
  - Manually in the variable editor (CTRL-C/V)
  - Import wizard
  - Programatically (repeatable)
    `textscan()`, `textread()`,
    `xlsread()`, `xlswrite()`, ...
Getting help

- **Console**
  - `help` command
  - `helpdesk`

- **On-line**
  - Detailed toolbox documentation,
  - function reference
  - and forums on [http://mathworks.com](http://mathworks.com)
Useful links

- MATLAB Help – Function reference, tutorials, videos

- MATLAB Central – Forums, file exchange
  - http://www.mathworks.com/matlabcentral

- Getting started with MATLAB
Questions?

THANKS FOR YOUR ATTENTION!