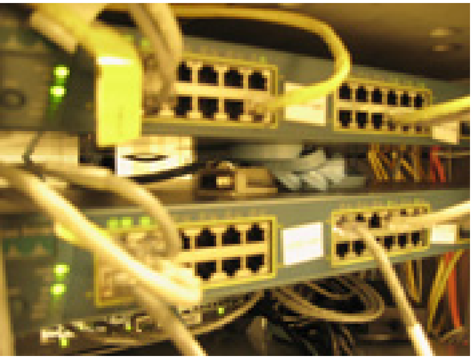


MatLab introduction

Risk analysis

8th Sep. 2014

Sipos Róbert
PhD candidate
BUTE Dept. of Networked Systems and Services
siposr@hit.bme.hu

2011. május 9.
Budapest

Agenda

- Introduction, architecture
- Data types
- Basic arithmetic operations and functions
- Random number generation
- Plotting
- Work with external data files
- Programming fundamentals
- „Hello, World!”

Introduction

- Cleve Moler (University of New Mexiko) 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
  - Some application requires symbolic calculations (for those use Mathematica for example)

Architecture

- Platform
  - Built-in functions
  - Toolboxes (e.g. neural networks, image and sound processing, financial, bioinformatics, …)
    - Both command line and GUI interfaces
  - Custom modules (programming language)
  - Data handling framework
    - File I/O
    - Database connections
- Advantages

  - Rapid prototype developing

  - No need to hassle with low level issues

  - Fast computation, optimized to the latest hardwares (e.g. GPGPU support)

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, …}
- Object oriented types
  - Strong connection with Java language
- Other types: e.g. strings, database connections

Basic operations (syntax)

- Defining constant values

  `A = [a11 a12; a21 a22];` (row-wise)

  ```
  row_vector = [v1 v2];
  column_vector = [v1; v2];
  ```

- Calling functions

  ```
  result = myfunc(0);
  [a, b, c] = myfunc(0);
  ```

  (optional number of return values)

- Indexing

  ```
  v2 = v(2); a12 = a(1, 2);
  ```

  - Starts from 1

Basic operations

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

Basic operations

- Transpose a matrix

```
a = a.';
```

`a = a';` (transpose & complex conjugate)

- Element-wise operations (dot)

```
c = a ./ b; % c(i) = a(i)/b(i);
```

- Series: `v = 0:2:200;`
- Select submatrices
  - `a = a(1:2, 1:2);`
  - `row1 = a(1, :);`
  - `column1 = a(:, 1);`
- Append: `v = [1 2]; v = [v 3];`

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 output the result to the console
- sin(), cos(), abs(), sqrt(), exp(), floor(), ceil(), log(), eig()
- min(), max(), mean(), sum(), prod(), std()
- ...

Handling complex values (*)

- Algebraic representation: `z = a + b*i;`
  - `a = real(z);`
  - `b = imag(z);`
- Trigonometric representation: `z = r * exp(fi*i);`
  - `r = abs(z);`
  - `fi = angle(z);`

Handling polinoms (*)

- Coefficient vectors

  e.g. $4x^4 + 3x^3 – x^2 + 1$ → `l = [4 3 -1 0 1];`

- Roots
  - `r = roots(l);`

- Characteristic polynom
  - `l = poly(r);` (normalized, first coeff. will be unit)

Random number generation

- Uniformly distributed pseudorandom numbers in the open interval (0,1)
  - `rand(n, [m])`
  - How to use it on an arbitrary interval?
- Normally distributed numbers (standard normal)
  - `randn(n, [m])`
  - How to use it with an arbitrary mean and std. deviation?
- Other distributions
  - `random(name, param1, [param2], n, [m])`
    - `'Binomial', n, p`
    - `'Exponential', mean`
    - `'Geomteric', p`
    - `'Poisson', lambda`
    - `'Normal', mean, stddev`
    - `..`

Plotting

- New plot window: `figure(n);`
- `plot(x, y, [x2, y2, …], [params]);`
  - x → D, y → R
  - Interpolation (default)
  - Only dots: `plot(x, y, '*');`
  - 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
  - Graphical editor
- File format: `.fig` (remains editable, stores the numerical data)
  - Also exportable to static image files of various formats

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 mathworks.com

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

Programming fundamentals

- Function declaration

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

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

Programming fundamentals

Conditionals

```
if expression1
   …]
[elseif expression2
   …]
[else
   …]
end
```

```
switch
   case expression1
      …
   [otherwise
      …]
end
```

Programming fundamentals

Loops

Count-controlled loop

```
v = 1:10;
for i = v
    …
end
```

Condition-controlled loop

```
while expression
   …
end
```

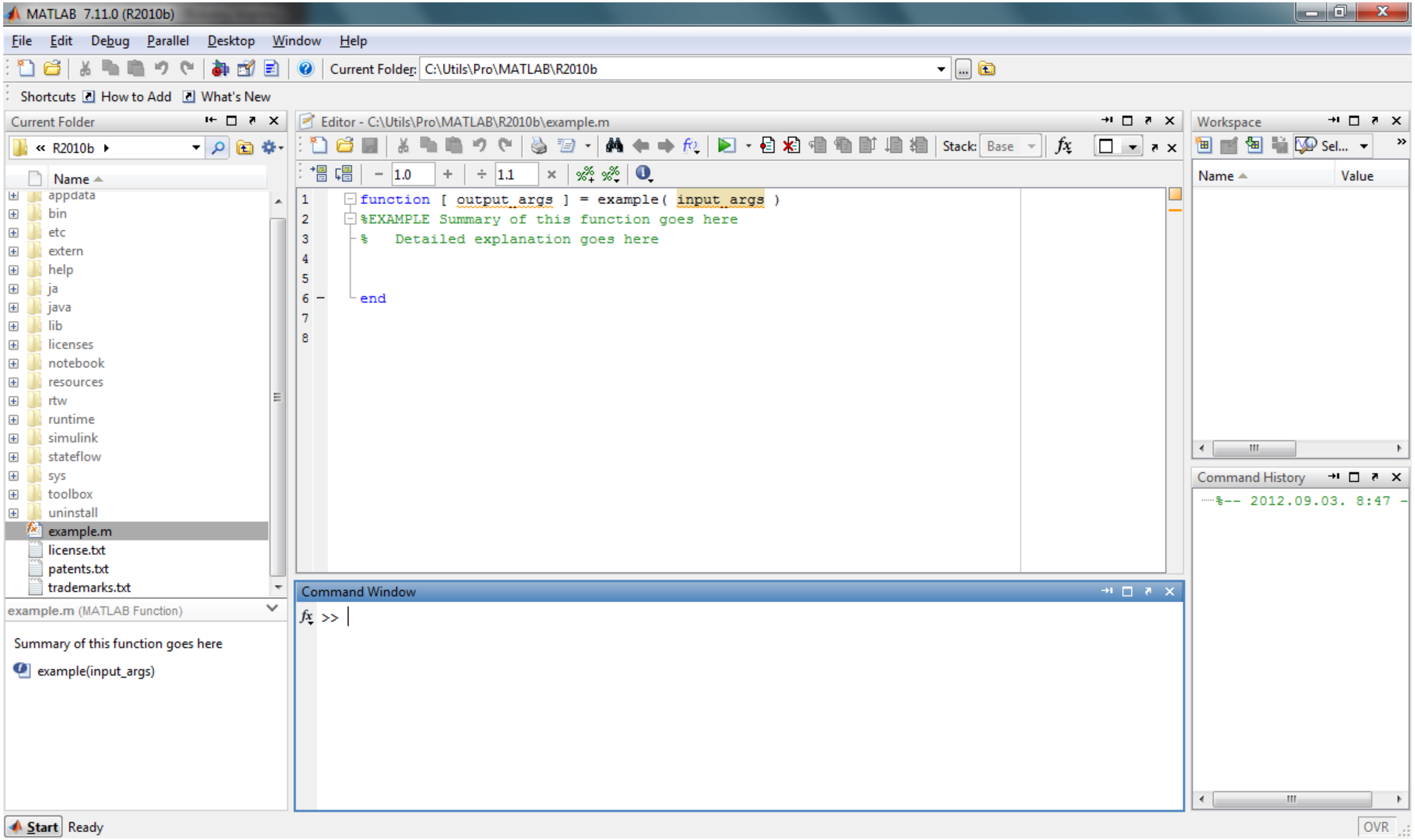
(no do-while loop)

Terminate the loop: `break;`
Continue with the next cycle: `continue;`

Further features (*)

- Handling strings
- Object oriented programming
- Using Java packages
- Creating GUI
- Using SQL databases
- ...

Window layout



Current Folder: C:\Utils\Pro\MATLAB\R2010b

« R2010b ▸

Name
appdata
bin
etc
extern
help
ja
java
lib
licenses
notebook
resources
rtw
runtime
simulink
stateflow
sys
toolbox
uninstall
example.m
license.txt
patents.txt
trademarks.txt

example.m (MATLAB Function)

Summary of this function goes here

example(input_args)

```
function [ output_args ] = example( input_args )
%EXAMPLE Summary of this function goes here
%   Detailed explanation goes here


end
```

```
%-- 2012.09.03. 8:47 -
```

```
>> |
```

„Hello, World!”

- Task #1
  - Create a new script (`lab1a.m`)
  - Generate `M` binary vectors using the random generator, containing `N` independently drawn element with `pi` probability of being 1
  - Display the vectors together with their probability
    Make a helper function: `function p = prob(y, pi)`

    Hint: `fprintf('format', v1, …)`
    `(→ %d %f \n \t)`
  - Run from console (`> lab1;`), check global variables
- Task #2 (`lab1b.m`)
  - Generate all possible binary vectors with `N` element
    (try to solve it if we need to store them, and if we don't)
  - Display them similarly

Useful links

MatLab function reference

`http://www.mathworks.com/help/techdoc/ref/f16-6011.html`

MatLab Central (forums, file exchange)

`http://www.mathworks.com/matlabcentral`

Kérdések?

KÖSZÖNÖM A FIGYELMET!

Híradástechnikai Tanszék