MATHEMATICAL AND COMPUTER MODELING OF NONLINEAR BIOSYSTEMS I

COMPUTER LABORATORY I: INTRODUCTION TO MATLAB®

Ph. D. Programme 2013/2014

PhD STUDIES
KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI
UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Project co-financed by European Union within the framework of European Social Fund

Workplan

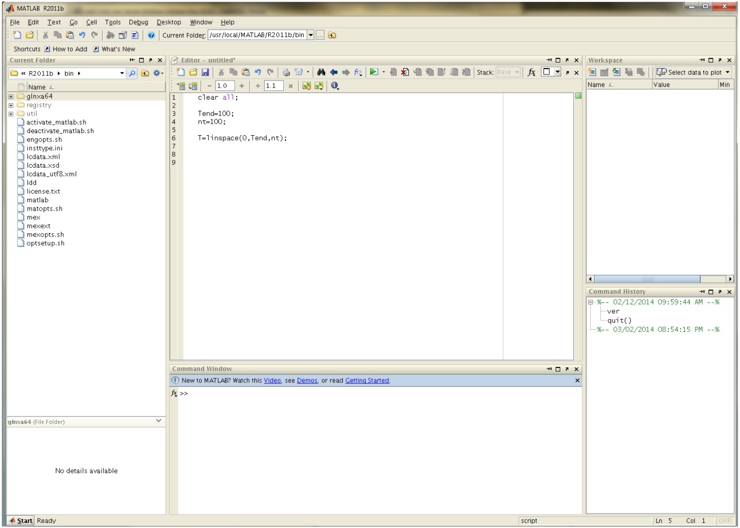
Workplan

1. MATLAB® basics — variable types, control structures, etc.;
2. solving ordinary differential equations in MATLAB®;
3. basic plotting.

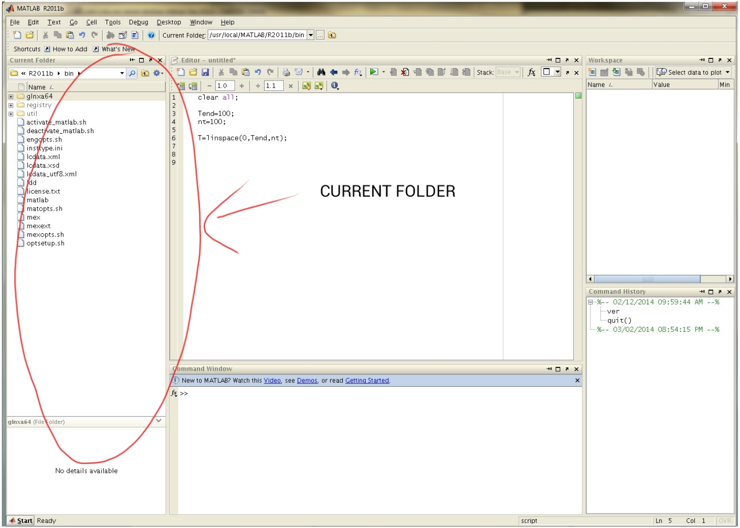
Today's specific aims

Writing MATLAB® script to solve continuous logistic equation.
Comparison of the numerical and analytical solutions.

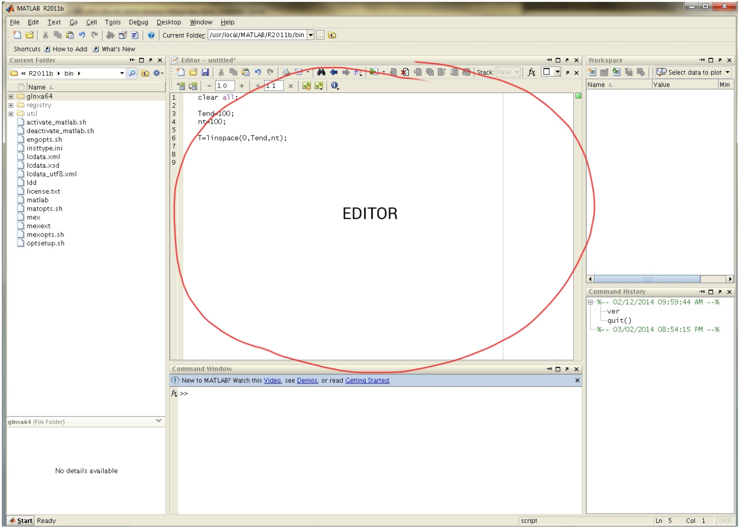
Interface overview



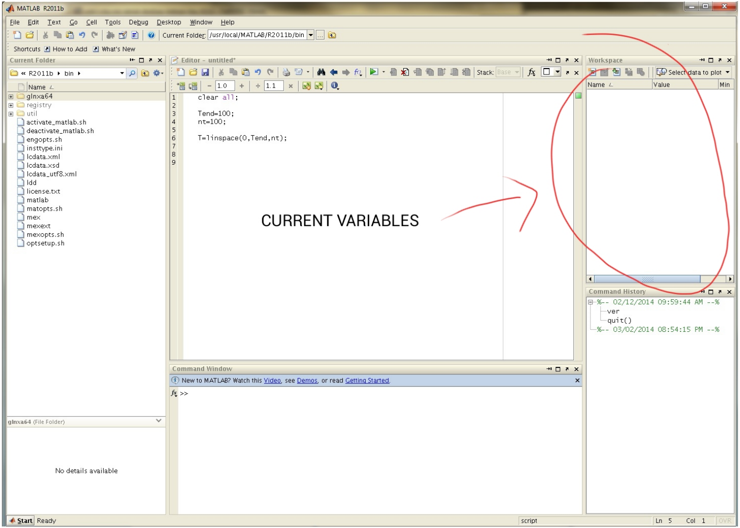
Interface overview



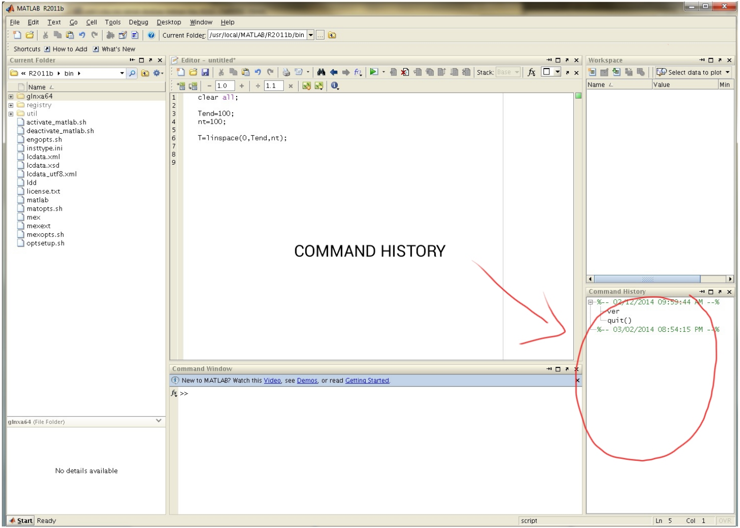
Interface overview



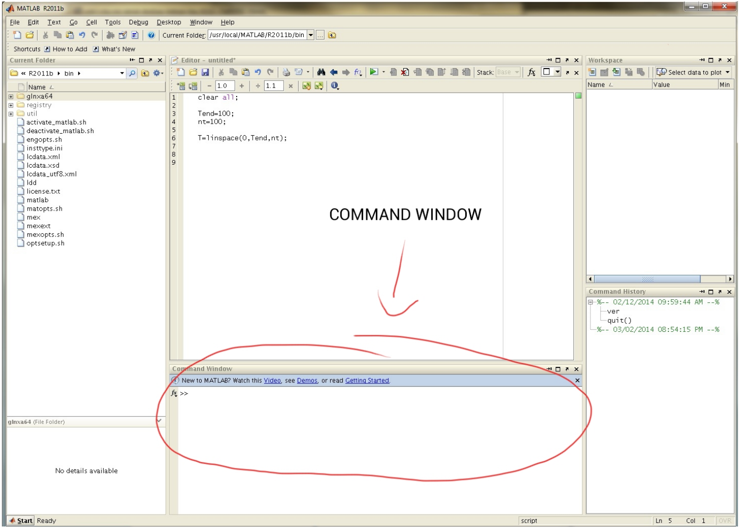
Interface overview



Interface overview



Interface overview



Basic Data Types

Numerical

```
a=2.3;
b=5.89e-04;
```

Vectors

```
a=[2, 3, 4, 5];
a=2:5;              %result: [2 3 4 5]
a=2:2:6;            %result: [2 4 6]
a=linspace(2,5,4);  %result: [2 3 4 5]
```

Matrices

```
a=[2, 3; 1, 2];
a=ones(2,3);      %2x3 matrix with ones
b=zeros(3,4,5);   %3x4x5 matrix with zeros
```

Basic Data Types

Strings

```
name = 'logistic';

i=1;
fileName = ['out' int2str(i) '.mat'];
```

Structure

```
model.name = 'logistic'
model.r     = 0.1;
model.K     = 10;

model.('name') = 'logistic2';
```

Basic Matrix Operations

Concatenating matrices

```
A=[1 2 3];
B=[4 5 6];
C=[A, B];
D=[A; B];
```

Accessing and assigning matrix elements

```
A(2)=4;
D(2,2)=3;
```

Accessing submatrix

```
D(:,[2 3])
D(2:end,2:end)
```

Basic Matrix Operations

Multiplication and division of matrices

```
A=[1 2];
B=[4 5];
C=A*B; C1=A^2;      %typical multiplication
D=A.*B; C1=A.^2;    %pair-waise operations
E=A./B;             %pair-waise operation
```

Element-wise functions

```
A=[2 3 4];
B=sin(A);
C=exp(A);
```

Find more by typing "doc elmat" — look into documentation.

Vectorization example

```
N=10000; %matrix size

A=rand(N,N);
B=zeros(size(A));

tic
for i=1:N
    for j=1:N
        B(i,j)=sin(A(i,j));     %evaluation time 5.83 sec
    end
end
toc

tic
    B=sin(A);                   %evaluation time 0.94 sec
toc
```

Basic Control Structures - Conditional clauses

IF

```
if a1<3
   a2=4;
elseif a1>4
   a2='jablko';
else
   a2=a2';
end
```

SWITCH

```
switch a1
   case 2
      a2='jablko';
   otherwise
      statements
end
```

Basic Control Structures - Loops

FOR

```
for i = 1:2:20
   A(i) = 2*i;
end
```

WHILE

```
while i<=20
   A(i) = 2*i;
   i=i+1;
end
```

Basic File Types

Program files can be:

- scripts — execute a series of MATLABMATLAB® statements;

```
clear all;
a1 = 3;
a2 = sin(a1);
```

- functions — accept input arguments and produce output.

```
function y=test(a1,a2)
     y=sin(a1)*a2;
```

Both scripts and functions contain MATLAB® code, and both are stored in text files with a .m extension (function name should be the same as the file name!).

Functions have their own workspace, separate from the base workspace.

- Good practice in complicated project is to have functions in separate folders.
- If you have some set of frequently used functions (your own functions) you can keep them in a fixed specific folder.
- In order to include function into current workspace you can use addpath function.

If you have your function in 'Additional functions' subfolder you should have

```
addpath('/Additional functions')
```

in your main script.

Exercise 1

Create MATLAB® function which takes matrix A as an argument and returns its maximal square sub-matrix. Place the function file in the 'Additional functions' folder.

Create MATLAB script in which the following :

1. create any two matrices: 3x4 A and 3x5 B;
2. create matrix C in which first 3 columns come from A and the rest come from B;
3. assign to the variable D the maximal square sub-matrix of C;
4. calculate sine of each element of E, where E is the matrix D with the opposite column order.

Exercise 1 - solution

Function returning maximal square submatrix (in file squareSubmatrix.m):

```
function B = squareSubmatrix(A)

    s=size(A); %s(1) - no. of rows
               %s(2) - no. of columns

    if s(1) >= s(2)
        B = A(1:s(2),:);
    else
        B = A(:,1:s(1));
    end

end
```

Exercise 1 - solution

Script (in file anyname.m):

```
clear all;

addpath('Additional functions\')

A=rand(3,4); B=ones(3,5);

C=[A(:,1:3) B];

D=squareSubmatrix(C);

sin(D(:,end:-1:1))
```

Excercise 2

Implement MATLAB® function taking the current populations size N and returning the derivative of the logistic equation:

$$\dot{N} = rN(1 - \frac{N}{K})\,.$$

Model parameters should be supplied as a structure.

Exercise 2 - solution

```
function dNdt = logisticRHS( N, par )

    dNdt=par.r*N*(1-N/par.K);

end
```

ODE solvers

Available solvers

RK methods: ode45, ode23

Adams: ode113

Trapezoidal rule: ode23t

Others: ode15s, ode23s, ode23tb, ode15i

ode45 in details

```
sol = solver(@odefun,[t_0 t_end],initial_condition,...
                options,...);
```

Output variable sol is a structure.

Excercise 3

Write MATLAB® script for solving logistic equation

$$\dot{N} = rN(1 - \frac{N}{K}) .$$

Exercise 3 - solution

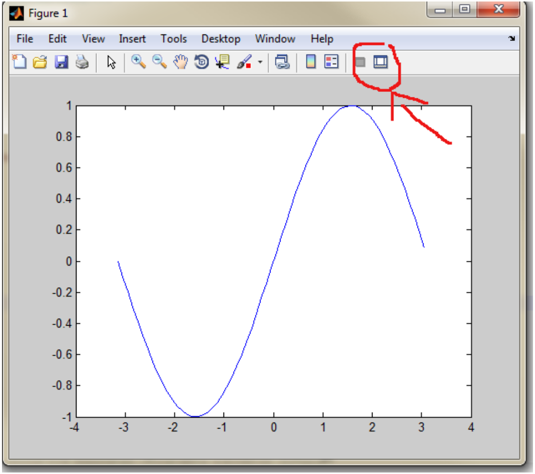
```
clear all;

N0=3; %initial condition
Tend=30; %end of the time span

params.r=0.1;
params.K=10;

sol = ode45(@logisticRHS,[0 Tend],N0,[],params);
```

Important: We need to modify the definition of logisticRHS function to

```
function dNdt = logisticRHS( t, N, par )
```

in order to use it with ode45 (add time dependence).

```
x = -pi:.1:pi;
y = sin(x);
plot(x,y)
```



Basic plot options

```
xlim([0 100]) %set limits for x-axis
ylim([0 10])  %set limits for y-axis

xlabel('Time(t)') %set x-axis label
ylabel('N(t)')    %set y-axis label

title('Logistic equation')  %plot title

grid on;
box on;
```

Excercise 4

Add plotting feature to the script from previous exercise (logistic equation solver).

Exercise 4 - solution

Add the following lines to the script from Ex. 3:

```
plot(sol.x,sol.y)

xlabel('Time(t)') %set x-axis label
ylabel('N(t)')    %set y-axis label

title('Solution to logistic equation')  %plot title

grid on;
box on;
```

Excercise 5

Knowing the exact solution to the logistic equation

$$N(t) = \frac{KN_0 e^{rt}}{K + N_0\left(e^{rt} - 1\right)}$$

Plot the error between the numerical and analytical solutions.

Exercise 5 - solution

Step 1. Implement function returning value of analytical formula:

```
function N = analyticalSol( t, N0, par )
    N=par.K*N0*exp(par.r*t)./(par.K+N0*(exp(par.r*t)-1));
end
```

Step 2. Add the following lines to the script from Ex. 5:

```
solA=analyticalSol(sol.x,N0,params);

plot(sol.x,solA-sol.y)
```

Excercise 6

Using the odeset function decrease solver tolerances (RelTol and AbsTol) to 1e-12 and plot the error between the numerical and analytical solutions.

How much longer did the calculations take?

Exercise 6 - solution

Modify the way in which we use ode45 function in the following way:

```
options=odeset('RelTol',1e-12,'AbsTol',1e-12);

sol = ode45(@logisticRHS,[0 Tend],N0,options,params);
```

To measure the time needed for evaluation of some code we use tic and toc:

```
tic

some code line 1...
some code line 2...

toc
```