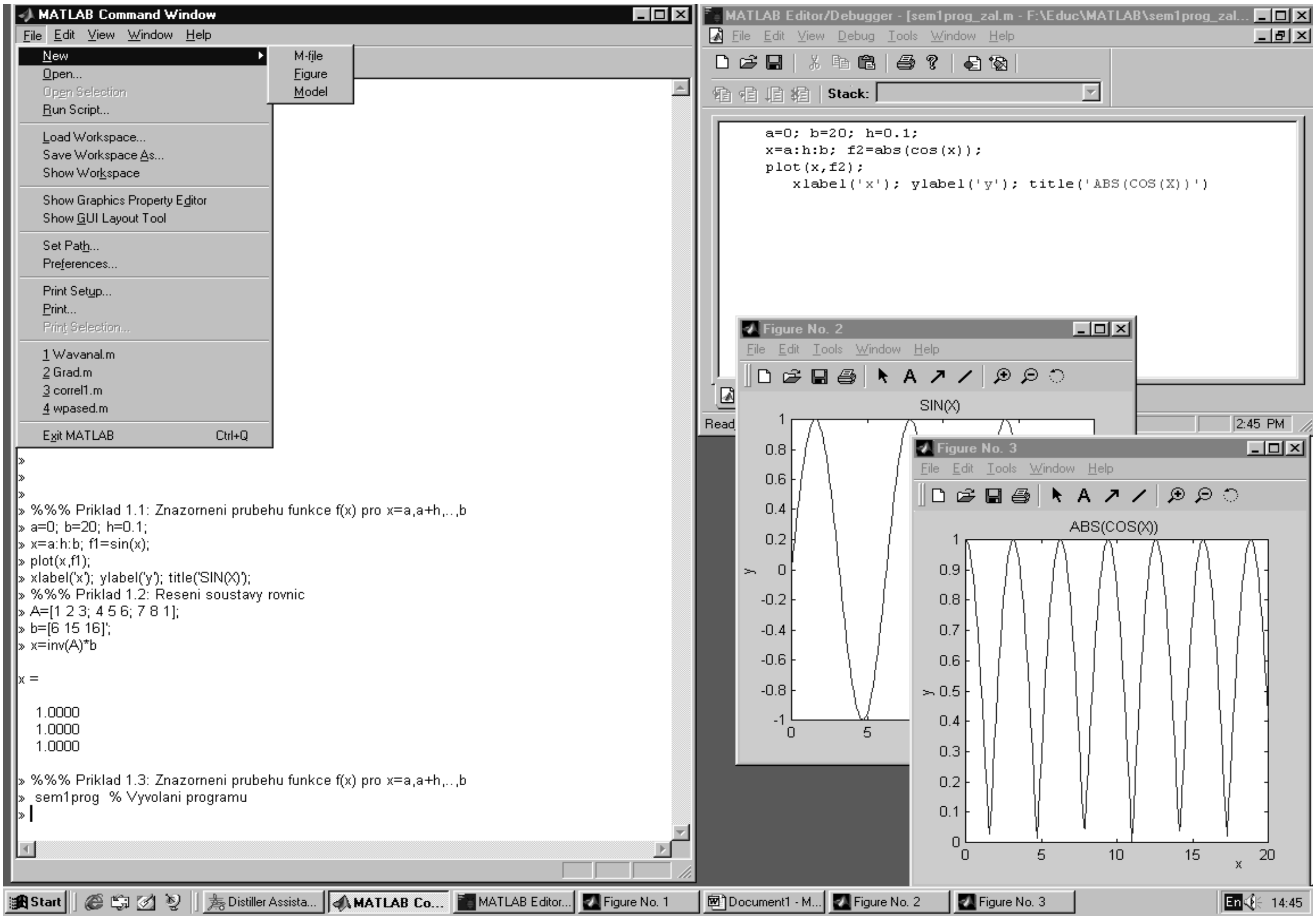
NUMERICAL ANALYSIS IN MATLAB

BASIC COMMANDS AND FUNCTIONS OF THE VIZUALIZATION AND PROGRAMMING ENVIRONMENT

Prof. Aleš Procházka
Institute of Chemical Technology, Prague
Department of Computing and Control Engineering
Technical Street 1905, 166 28 Prague 6

CONTENTS



Textbook of Mathematical Methods in Engineering
Akademic Year 2005/2006
Prof. Ing. Aleš Procházka, CSc

1. MATLAB ENVIRONMENT

Characteristics

- Matrix oriented computing and graphical environment
- COMMAND window, FIGURE window, EDIT window
- Conversational and programming style, environment initialization
- Possibilities of data export and import, the use of toolboxes, simulation

1.1 Conversational Computational Mode

Notes to basic commands:

1. Command ended by a semicolon $\Longrightarrow$ store of a variable without its visualization
2. Return to previously written commands $\Longrightarrow$ up-arrow key
3. Help (help, help ⟨*command*⟩, lookfor ⟨*key*⟩, ...) and DOS commands (dir,...)
4. Visualization of variables (whos), delete of variables (clear ⟨*list*⟩)
5. The use of standard variables: pi, eps, i, j

```
%%% Example 1.1: Plot of a function f(x) for x=a,a+h,..,b
>>  a=0; b=20; h=0.1;
>>  x=a:h:b; f1=sin(x);
>>  plot(x,f1);
>>  xlabel('x'); ylabel('y'); title('SIN(X)');
```

1.2 Matrix and Vector Operations

Basics of the work with matrices:

1. The necessity of the use of correct dimensions of matrices and vectors
2. Selected operations (transposition, multiplication, inversion)

```
%%% Example 1.2: Solution of the set of linear equations A x = b
>>  A=[1 2 3; 4 5 6; 7 8 1];
>>  b=[6 15 16]';
>>  x=inv(A)*b
```

1.3 Programming Computational Mode

The style of work in EDIT and COMMAND window:

1. Opening of the EDIT window
2. Selection of commands - programming
3. Save of the final programme under a selected name
4. Start of the programme from the COMMAND window

```
%%% Example 1.3: Plot of a function f(x) for x=a,a+h,..,b
>>  sem1prog  % Programme call
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% SCRIPT SEM1PROG.M %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  a=0; b=20; h=0.1;
  x=a:h:b; f2=abs(cos(x));
  plot(x,f2);
     xlabel('x'); ylabel('y'); title('ABS(COS(X))')
```

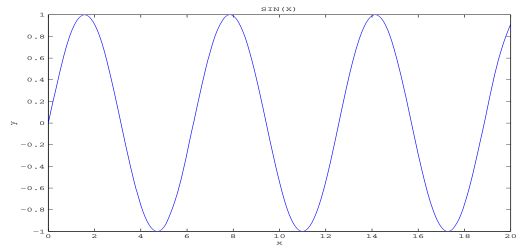
COMMANDS

PLOT
XLABEL
YLABEL
TITLE

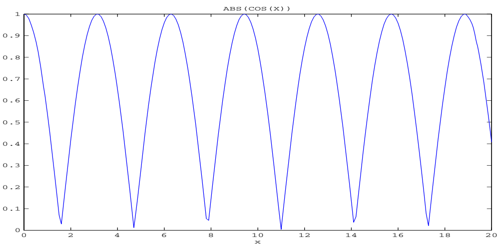
SIN
COS
ABS

INV

Solution of Examples



$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 6 \\ 15 \\ 16 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$



EXAMPLES 1

1.1 Evaluate and plot function f(x)=exp(x) for $x \in \langle a, b \rangle$ and step h (a=0, b=2, h=0.2)

1.2 Evaluate and plot function f(x)=log(x) for $x \in \langle a, b \rangle$ and step h (a=0.1, b=5, h=0.1)

1.3 Solve the system of equations A x = b pro A=[1 2; 3 3]; b=[5; 7];

2. BASIC OPERATIONS

Operators

- arithmetic: matrix: `+,-,*,/,^,'`
  vector: `,.*,./,.^,.'` (element-by-element operations)
- relational: `>,>=,==,=~,<,<=`
- logical: `&,|,~`

COMMANDS

EXP
SUM
MIN
MAX

DET

ONES
ZEROS

RAND

2.1 Assignment Commands

$\langle variable \rangle = \langle expression \rangle$

```
%%% Example 2.1: Operations with vectors
>>  v=[1 2 3 4];
>>  r1=v*v'
>>  r2=v.*v
>>  r3=v'*ones(1,4)
%%%
%%% Example 2.2: Relational expressions
>>  s1=3>5
>>  s2=3<5
%%% Example 2.3: Matrix operations
>>  A=[1 2; 3 4]
>>  b=[1 2]
>>  C=[A, b';[1 1 1]]
```

Solution

$$v = \begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix}$$

$$r1 = 30$$

$$r2 = \begin{pmatrix} 1 & 4 & 9 & 16 \end{pmatrix}$$

$$r3 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{pmatrix}$$

$$s1 = 0, \quad s2 = 1$$

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad b = \begin{pmatrix} 1 & 2 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 2 & 1 \\ 3 & 4 & 2 \\ 1 & 1 & 1 \end{pmatrix}$$

2.2 Functions

Function categories:

1. Scalar: SIN, COS, EXP, LOG, ...
   $\Longrightarrow$ function is applied to each matrix element
2. Vector: SUM, MIN, MAX, MEAN, STD, ...
   $\Longrightarrow$ function is applied for each matrix column
3. Matrix: INV, DET
   $\Longrightarrow$ function is applied for the whole matrix

Solution

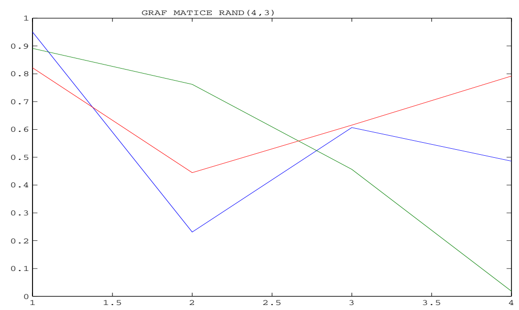
$$A = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$$

$$f1 = \begin{pmatrix} 0.8415 & 0.9093 \\ -0.7568 & -0.9589 \end{pmatrix}$$

$$f2 = \begin{pmatrix} 5 & 7 \end{pmatrix}$$

$$f3 = -3$$

```
%%% Example 2.4: Functions
>>  A=[1 2;4 5];
>>  f1=sin(A)   % scalar function
>>  f2=sum(A)   % vector function
>>  f3=det(A)   % matrix function
%%%
%%% Example 2.5: Special functions
>>  z1=zeros(2,3)
>>  z2=ones(1,3)
>>  z3=rand(4,3)
>>  plot(z3);  title('PLOT OF MATRIX R=RAND(4,3)')
```



EXAMPLES 2

2.1 Define matrix A=[1 1.1 1.2;1.5 1.7 1.9; 2.1 2.4 2.7] and evaluate
- mean values of its rows and columns
- minimum and maximum values of the whole matrix
- determinant of the given matrix

2.2 Apply function HIST for the study of distribution of values v=RAND(1,N) assuming their number N=100, 500, 1000. Plot resulting random values by function PLOT (apply key ZOOM IN as well)

3. CONTROL COMMANDS

3.1 Loop Commands

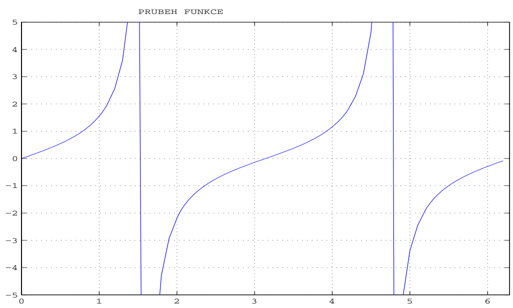
for $\langle variable \rangle = \langle expression \rangle$
    $\langle commands \rangle$
end

```
%%% Example 3.1: Evaluate the approximate value of integral of function
%%%   f(x)=sin(x) in limits a=0, b=20 for N=20 its parts
%%%   using rectangular rule:
%%%     Q ~ h*sum(f(x(i)) pro x(i)=a+(i-1)*h, i=1,2,...,N
%%%
    a=0; b=20; N=20; h=(b-a)/N;  % Definition of given values
    x=a:h:b; f=sin(x);           % Evaluation of function values
% Standard algorithm for the evaluation of the sum of given values
    S=0;
    for i=1:N;
       S=S+f(i);
    end;
    Q1=S*h
% Compressed algorithm for the evaluation of the sum of given values
    Q2=sum(f(1:N))*h
```

3.2 Decision Structures

if $\langle relational\_expression\_1 \rangle$
    $\langle commands\_1 \rangle$
elseif $\langle relational\_expression\_2 \rangle$
    $\langle commands\_2 \rangle$
else
    $\langle commands\_3 \rangle$
end

```
%%% Example 3.2: Plot of the given function in selected limits
%%%
    a=0; b=2*pi; h=0.1; x=a:h:b;
    k=menu('FUNCTION','Sin(x)','Cos(x)','Tan(x)');
    if k==1
      plot(x,sin(x))
    elseif k==2
      plot(x,cos(x))
    else
      plot(x,tan(x)); axis([0 2*pi -5 5]);
    end
    grid on
```



COMMANDS

FOR-END
IF-ELSEIF-ELSE-IF
MENU
FIGURE
AXIS
GRID

EXAMPLES 3

3.1 Evaluate approximate value of the integral of function $f(x)=sin(x)$ in limits $a=0$, $b=\pi$ for $N=20$, 40 and 60 its parts using the trapezoidal rule

3.2 Enlarge the algorithm resulting from the previous example by the MENU command for selection of variable N

4. SUBMATRICES

4.1 Submatrices

```
%%% Example 4.1: Basic matrix operations
>>  A=[1 2 3; 4 5 6; 7 8 9];
>>  B=A(2,:)
>>  C=A(:,[1 3])
>>  D=A(:,[3:-1:1])
```

4.2 Logical Operations

```
%%% Example 4.2: Logical matrix operations
>>  L=A(:,3)>8, A1=A(L,:)
>>  v=A(:)'
```

Solution

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$$B = \begin{pmatrix} 4 & 5 & 6 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 3 \\ 4 & 6 \\ 7 & 9 \end{pmatrix} \qquad D = \begin{pmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{pmatrix}$$

$$L = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \qquad A1 = \begin{pmatrix} 7 & 8 & 9 \end{pmatrix}$$

$$v = \begin{pmatrix} 1 & 4 & 7 & 2 & 5 & 8 & 3 & 6 & 9 \end{pmatrix}$$

4.3 Function Subroutines

function [$\langle output\_parameters \rangle$]=$\langle name \rangle$($\langle input\_parameters \rangle$)
    $\langle commands \rangle$

Notes:

1. Formal parameters: matrices, vectors, strings
2. The number of real parameters can be smaller than that of formal parameters

```
%%% Example 4.3: Evaluate the approximate value of integral
%%%   of function f(x)=sin(x) in limits a=0, b=20 for
%%%   N=20 its parts using rectangular rule:
%%%     Q ~ h*sum(f(x(i)) pro x(i)=a+(i-1)*h, i=1,2,...,N
%%%
      a=0; b=20; N=20;
      Q=sem4prog(a,b,N)
```

COMMANDS

:
CLC
BREAK
FUNCTION

EXAMPLES 4

4.1 Define function subroutine for evaluation of minimum and maximum value of the given matrix and apply it for matrix $\mathbf{A} = [1\ 3\ 5;\ 2\ 4\ 6;\ 1\ 1\ 1]$

4.2 Modify the given vector $\mathbf{x}$=[1.1 1.4 1.9 2.1 1.3 1.7 1.8 5 1.5] by elimination of values that differ by more than the standard deviation from the mean value of the given sequence

5. OBJECTS AND 2D GRAPHICS

5.1 Object Properties

$$\langle name \rangle = \langle command \rangle \; (\; \langle property\_name \rangle, \; \langle property\_value \rangle \;)$$

```
%%% Example 5.1: Definition of figure properties, axes properties,
%%% and their modification
%%%
    figure;  set(gcf,'Units','Normal','Position',[0.5,0.57,0.3,0.35]);
    h1=axes('Position',[0.1 0.1 0.8 0.3]);
    plot(sin(0:0.1:29))
    get(h1)
%%%
%%% Definition of a property
  set(h1,'Units','Normal','Position',[0.5 0.5 0.3 0.3]);
```

COMMANDS

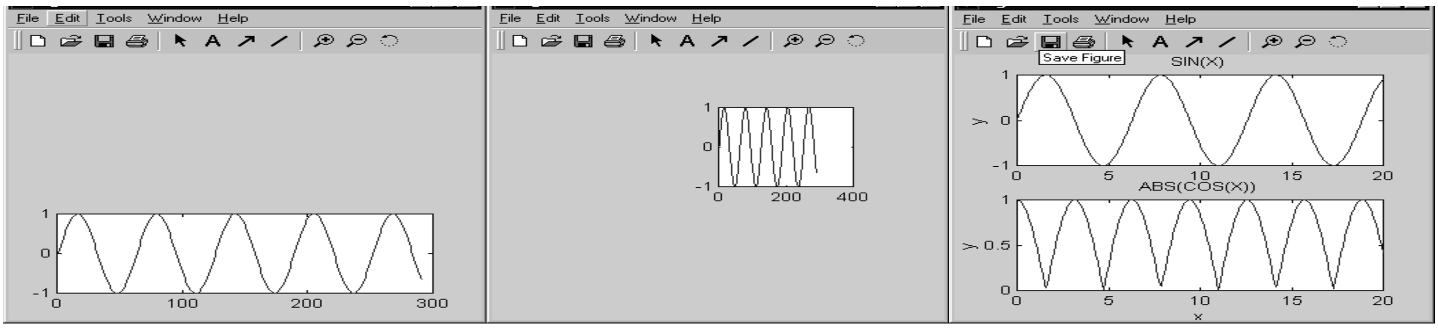
PLOT
SUBPLOT
XLABEL
YLABEL
TITLE
TEXT

AXIS, GRID
SET, GET
DELETE
HOLD ON
HOLD OFF

GINPUT

5.2 Two-dimensional Graphics

```
%%% Example 5.2: Plot functions f1(x)=sin(x) and
%%%   f2(x)=abs(cos(x)) for x=a:h:b choosing a=0,
%%%   b=20 and h=0.1 in two separate windows
%%%
    a=0; b=20; h=0.1;
    x=a:h:b;
    figure;  set(gcf,'Units','Normal','Position',[0.53,0.54,0.3,0.35]);
      subplot(2,1,1); plot(x,sin(x));
        ylabel('y'); title('SIN(X)');
      subplot(2,1,2); plot(x,abs(cos(x)));
      xlabel('x'); ylabel('y'); title('ABS(COS(X))');
```



EXAMPLES 5

5.1 Plot function $f(x) = sin(x)^2$ and its the first and second derivative for $x \in \langle a, b \rangle$ for $a = 0$, $b = 20$ in three separate figures

5.2 Plot $f(x) = cos(x)$ and its derivative in one figure window axes using different colors for each function

5.3 Use GINPUT command for estimation of the smallest positive root of equation: $sin(x) - 0.1x = 0$

6. 3D GRAPHICS

6.1 Fundamentas of 3D Graphics

Notes to the use of basic commands:

1. Definition of x-axis and y-axis values are defined by corresponding values of matrices **X**, **Y**
   $\Longrightarrow$ function MESHGRID
2. Selection of 3D plotting $\Longrightarrow$ function MESH, MESHC, ...
3. Selection of the viewpoint $\Longrightarrow$ function VIEW
4. The choice of the number of contour lines and their description $\Longrightarrow$ funkce CONTOUR

```
%%% Example 6.1: Plot function
%%%   f(x,y)=-x exp(-x^2-y^2) for x=-2:0.2:2 and y=-2:0.2:2
%%%   together with corresponding contour lines
%%%
    [X,Y]=meshgrid(-2:0.2:2);
    Z=-X.*exp(-X.^2-Y.^2);
      meshc(X,Y,Z);
      title('f(x,y)=-x*exp(-x^2-y^2)'); xlabel('x'); ylabel('y');
      colormap([0 0 1])
```

COMMANDS

MESHGRID
MESH
MESHC
GRID ON
GRID OFF
LINE
VIEW

HOLD ON
HOLD OFF

CONTOUR

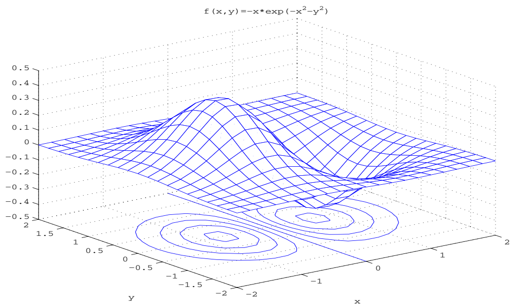
COLORMAP
COLORBAR

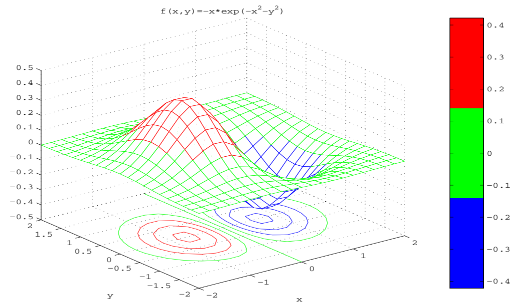
6.2 Colors

Notes to the use of basic commands:

1. Palette of m colors is defined by values of matrix $\mathbf{M}_{m,3}$
2. Each of m colors is defined by the combination of 3 basic colors (**R**ed-**G**reen-**B**lue) and their intensity in the range of $\langle 0, 1\rangle$
3. Application of the color map $\Longrightarrow$ function COLORMAP
4. Plot of the color map $\Longrightarrow$ function COLORBAR

```
%%% Example 6.2: Store the present colormap into matrix M1,
%%%   define a new colormap M2 consisting of three basic colors
%%%   and apply it to the given 3D surface
%%%
  M1=colormap
  M2=[0 0 1; 0 1 0;1 0 0]
  colormap(M2)
  colorbar
```





EXAMPLES 6

6.1 Plot function $f(x, y) = -x\ exp(-x^2 - y^2)$ using four selected colors and use a separate window to plot its contour plot for $x \in \langle -2, 2\rangle$ and $y \in \langle -2.5, 2.5\rangle$

6.2 Plot function $f(x, y) = -x^2y^2$ for $x \in \langle -2, 2\rangle$ and $y \in \langle -2, 2\rangle$

6.3 Visualize contour plot of function $f(x, y) = -x^2y^2$ for $x \in \langle -2, 2\rangle$ and $y \in \langle -2, 2\rangle$ with line description

7. DATA FILES MANIPULATION

COMMANDS

SAVE
LOAD
WHOS
DIR

DDEINIT
DDEREQ

7.1 Data Saving

save $\langle file\_name \rangle$ [$\langle list\_of\_variables \rangle$] [-ascii]

Notes to data saving:
1. In case that the list_of_variables is not defined the whole workspace is saved
2. Parameter $-ascii$ allows data store as a plain text (without its name)

```
%%% Example 7.1: Saving of variables
    A=[1 2 3; 4 5 6; 7 8 1];  b=[6 15 16]'; save dat1 A b;
```

7.2 Data Retrieving

load $\langle file\_name \rangle$

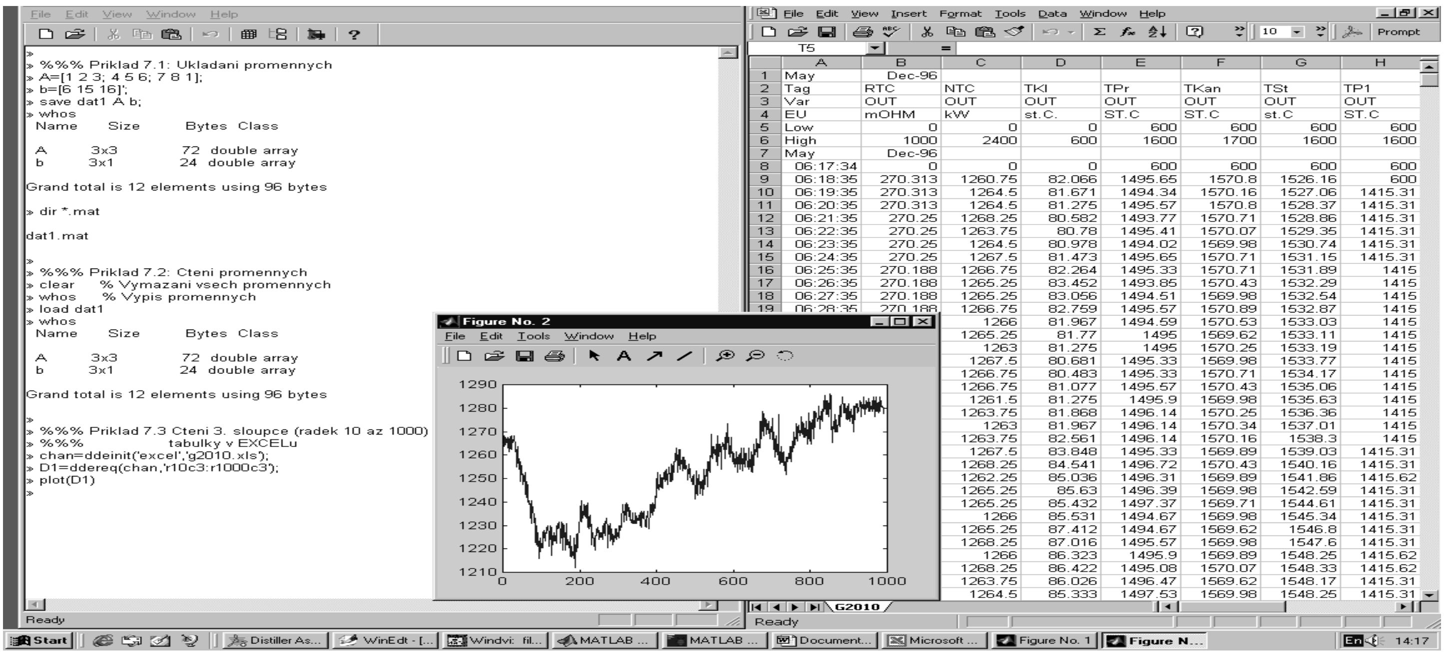
Notes to the data retrieving:
1. In case of the binary file (with MAT extension) all variables are retrieved together with their names
2. In case of the plain text file all its values are retrieved under the file_name

```
%%% Example 7.2: Data retrieving
    load dat1
```

7.3 Data Import from EXCEL

Steps:
1. Opening of the MATLAB environment and EXCEL $\langle file\_neme \rangle$.xls
2. Data transport from EXCEL to MATLAB: chan=ddeinit('excel','$\langle file\_name \rangle$.xls');
   D1=ddereq(chan,'r$\langle r1 \rangle$c$\langle c1 \rangle$:r$\langle r2 \rangle$c$\langle c2 \rangle$');
   where r1,r2,c1,c2 stand for initial and final number of rows and columns

```
%%% Example 7.3: Reading of the EXCEL table
    chan=ddeinit('excel','g2010.xls');
    D1=ddereq(chan,'r10c3:r1000c3'); plot(D1)
```



EXAMPLES 7

7.1 Using table G2010.XLS import into the MATLAB environment the power consumption (the 3rd column) and channel temperature in the glass furnace (the 6th column), plot these variables and evaluate their mean values

7.2 Using table G2010.XLS import into the MATLAB environment the matrix of measured values from the second up to the seventh column, save them into the *.MAT file and plot these variables in chosen limits

8. GRAPHICAL USER INTERPHASE

8.1 Steps of GUI Construction

1. Opening of the environment $\Longrightarrow$ File / New / GUI
2. Either opening of a new environment or existing figure
3. Opening of the PROPERTY INSPECTOR for each object and property
4. Opening EDITOR and modification of commands

```
%%% Example 8.1: Plot of a selected function
%%% Example 8.1: Plot of a harmonic function with its frequency modified by
```

8.2 Menu Creation

1. Opening of the environment $\Longrightarrow$ Tools / MenuEditor
2. Creation of the menu structure
3. Modification of related functions in the EDITOR environment

8.3 Hidden Functions Use

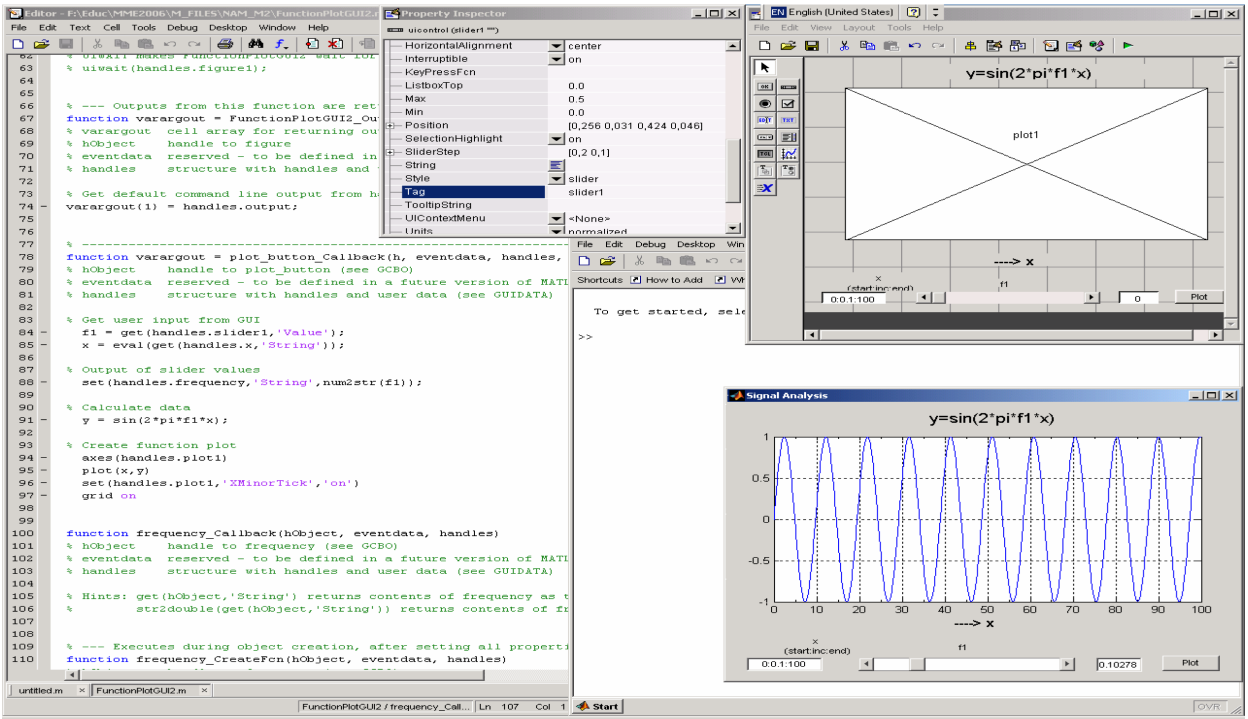
1. Modification of callback functions for specific object in the EDITOR environment

Note: Object handles are available for all functions

OBJECTS

PUSH BUT
SLIDER
RADIO BUT
CHECK BOX
EDIT TEXT
STATIC TEXT

MENU
AXES

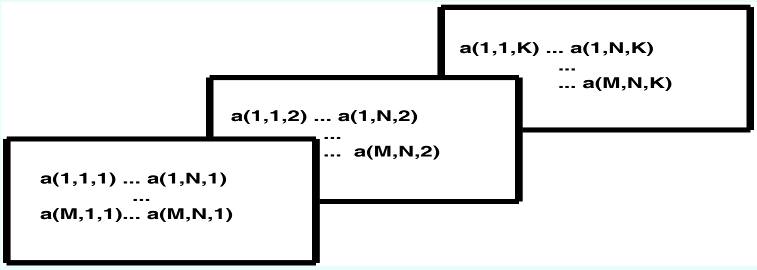


EXAMPLES 8

8.1 Modification of slider limit values
8.2 Modification of plotted function

9. MULTIDIMENSIONAL ARRAYS

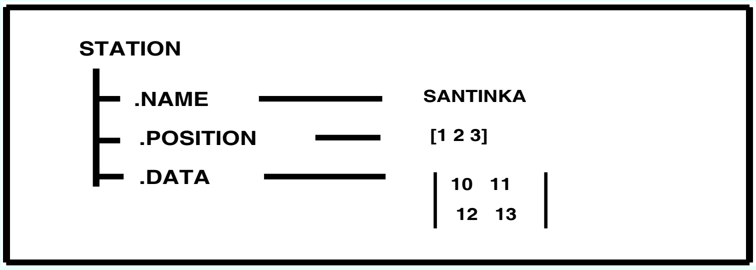
9.1 Multidimensional Arrays



1. The number of levels is not limited
2. Commands for multidimensional data processing are the same as that for matrices

```
%%% Example 9.1: Multidimensional array definition
>> A(2,2,1)=1;
>> A(2,2,2)=2;
>> A      % Variable display
```

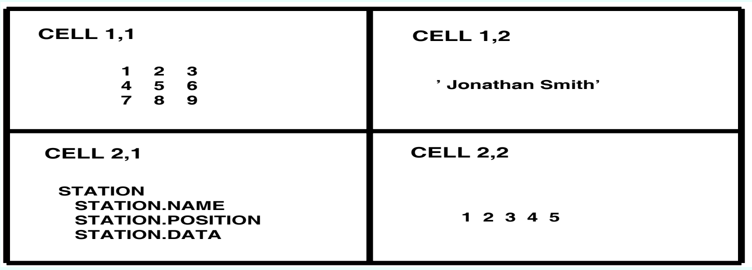
9.2 Structured Arrays



1. The system enables construction of simple databases
2. Data items can be of the different kind

```
%%% Example 9.2: Structured array definition
>> STATION(1).NAME='SANTINKA';
>> STATION(1).POSITION=[1 2 3];
>> STATION(1).DATA=[10 11;12 13];
>> STATION             % Variable display
>> D=STATION(1).DATA   % Data selection
```

9.3 Cells



1. Data items can be of any kind

```
%%% Example 9.3: The field of cells
>> C{1,1}=[1 2 3; 4 5 6; 8 9 10];
>> C{1,2}='Jonathan Smith';
>> C{2,1}=STATION;
>> C{2,2}=[1 2 3 4 5]; C  % Variable display
>> P=C{2,1}.POSITION(2)    % Data selection
```

COMMANDS

WHOS

EXAMPLES 9

9.1 Using table G2010.XLS import into the MATLAB environment temperatures measured at different parts of the glass furnace (in columns 4, 5 and 6) and store them in the structured array

10. SYMBOLIC MATHEMATICS

10.1 Symbolic Manipulation

```
%%% Example 9.1: Definition and manipulation with symbolic variables
  >>  syms a b c x n t p;
  >>  y1=3*x^2+5*x+3; y2=5*x^2+6; y3=y1+y2; pretty(y3)
  >>  y4=sin(x)^2+cos(x)^2; y5=simplify(y4)
  >>  ezplot(y3,[-2 1])
```

10.2 Symbolic Substitution

```
%%% Example 10.2: Substitution and visualization
  >>  d=(a+b)^(1/2);
  >>  f1=subs(d,a,0)
  >>  f2=subs(d,{a,b},{1 4})
```

10.3 Symbolic Summation

```
%%% Example 10.3: Evaluation of sum(1/n^2) for n=1,2,...
  >>  s1=symsum(1/n^2,1,inf)
```

10.4 Symbolic Differentiation and Integration

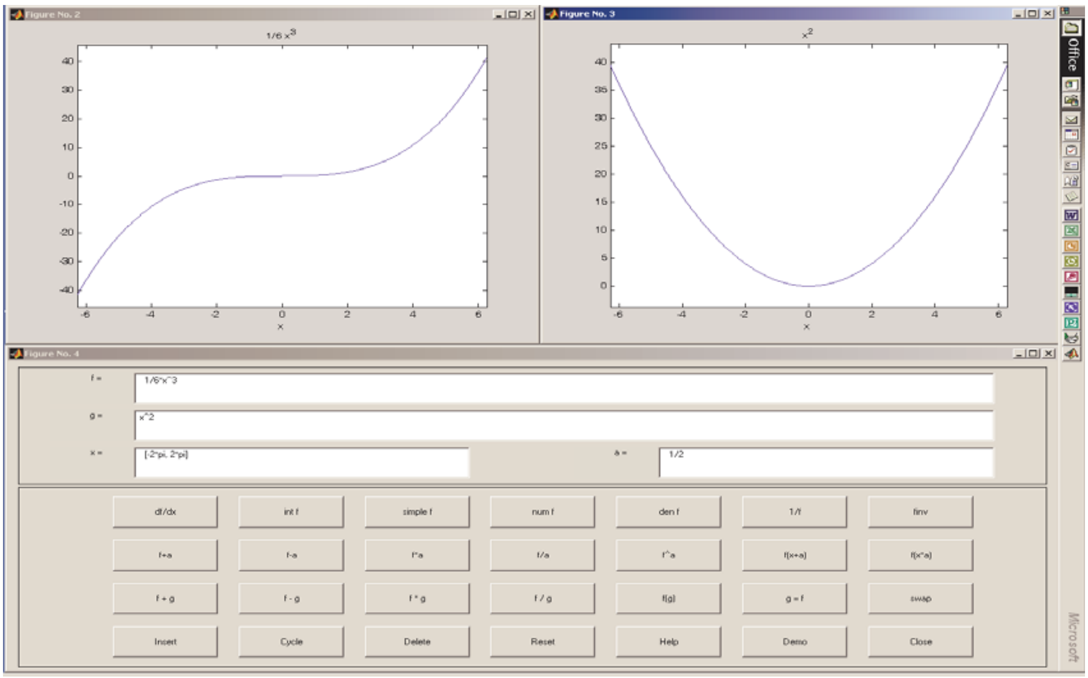
```
%%% Example 10.4: Substitution and visualization
  >>  f1=sin(a*x+b)^2; df=diff(f1)
  >>  f2=1/x; intf2=int(f2)
```

COMMANDS

SYMS
PRETTY
SIMPLIFY
SUBS

SYMSUM
DIFF

EZPLOT



EXAMPLES 10

10.1 Plot selected symbolically defined functions
10.2 Evaluate chosen sums of given sequences

11. SIMULINK ENVIRONMENT

11.1 Steps of Simulink Modelling

1. Opening of block library and model window $\Longrightarrow$ model composition
2. Definition of block values
3. Definition of modelling parameters and start of modelling

```
%%% Example 11.1: Visualization of a harmonic function
```

11.2 Applications

1. Solution of differential equations and continuous signal modelling
2. Time series and matrix manipulation resulting in discrete signals modelling
3. Real data acquisition using applications libraries and submodels definition

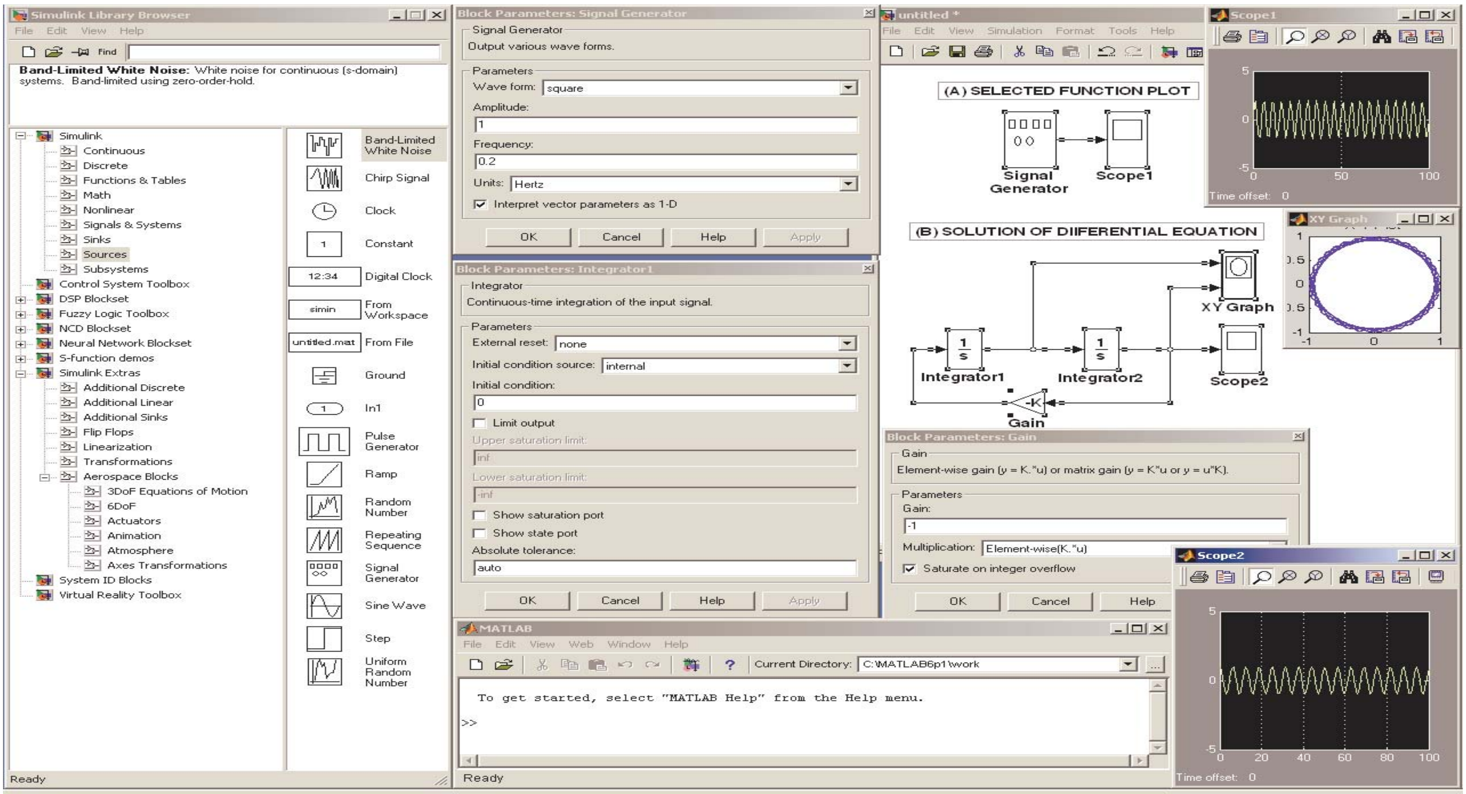
```
%%% Example 11.2: Solution of differential equation: y''+y=0, y(0)=0, y'(0)=1
%%% Solution: 1. Explicit definition of the highest derivative: y''=-y
%%%           2. Integration and corresponding block definition: y'=int(-y) dy
%%%           3. Description of another integration block: y=int(y') dy
```

BLOCKS

SIGNAL GEN
SINE
CONSTANT
GAIN

SCOPE
XY GRAPH
DISPLAY

FROM/TO WORKSPACE



EXAMPLES 11

11.1 Visualization of selected functions
11.2 Solution of given differential equation

12. SYSTEMS OF LINEAR EQUATIONS

Problem statement: Solution of system of equations

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ & & \cdots & \\ a_{N,1} & a_{N,2} & \cdots & a_{N,N} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \cdots \\ b_N \end{pmatrix} \quad \Leftrightarrow \quad \mathbf{A}_{N,N}\, \mathbf{x}_{N,1} = \mathbf{b}_{N,1}$$

12.1 Symbolic Solution

Characteristics:

1. Possibilities of simplification of symbolic solution
2. Substitution allows conversion to numerical solution

```
%%% Example 12.1: Symbolic solution of linear equations
>>    syms a11 a12 a21 a22 b1 b2 x1 x2    % Definition of symbolic variables
>>    A=[a11 a12; a21 a22]; b=[b1; b2]; x=A\b; pretty(simple(x))
%%% Alternative
>>    EQ=A*[x1; x2]-b; SOLUTION=solve(EQ(1),EQ(2));
>>      s1=SOLUTION.x1; pretty(simple(s1))
>>      s2=SOLUTION.x2; pretty(simple(s2))
>>    s=subs(x,{a11 a12 a21 a22 b1 b2},{1 1 2 1 3 4}) % Substitution
```

COMMANDS

SYMS
SOLVE
SUBS

SIMPLE
PRETTY

INV
DIAG

SPARSE
FULL
SPY
GPLOT

12.2 Numerical Solution

12.2.1 Finite Methods

Mathematical background of Gauss-Jordan method:

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} & b_1 \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} & b_2 \\ & & \cdots & & \\ a_{N,1} & a_{N,2} & \cdots & a_{N,N} & b_N \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & \cdots & 0 & x_1 \\ 0 & 1 & \cdots & 0 & x_2 \\ & & \cdots & & \\ 0 & 0 & \cdots & 1 & x_N \end{pmatrix}$$

System definition

$$\mathbf{A} = \begin{pmatrix} 4 & -1 & 1 \\ 1 & 6 & 2 \\ -1 & -2 & 5 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 4 \\ 9 \\ 2 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

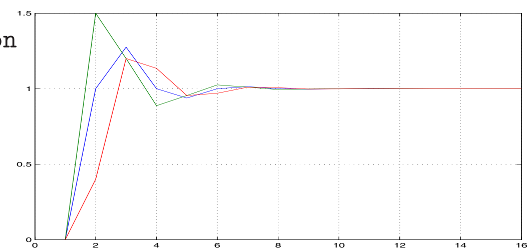
```
%%% Example 12.2: Gauss-Jordan method
>>  A=[4 -1 1;1 6 2;-1 -2 5]; b=[4 9 2]'; S=[A,b]; [m,n]=size(S);
>>  for i=1:m
>>    [Max,j]=max(abs(S(:,i)));  % Localization of the maximum element of i-th column
>>    S([j i],:)=S([i j],:);     % Row exchange
>>    S(i,:)=S(i,:)/S(i,i);      % Division of the i-th row by the diagonal element
>>    for ii=[1:i-1,i+1:m]       % Elimination
>>      S(ii,:)=S(ii,:)-S(ii,i)*S(i,:);
>>    end
>>  end; x=S(:,4)
```

12.2.2 Iterative Methods

Mathematical background:

$$\begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \cdots \\ x_N^{(k)} \end{pmatrix} = \left(\begin{pmatrix} b_1 \\ b_2 \\ \cdots \\ b_N \end{pmatrix} - \begin{pmatrix} 0 & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & 0 & \cdots & a_{2,N} \\ & & \cdots & \\ a_{N,1} & a_{N,2} & \cdots & 0 \end{pmatrix} \begin{pmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ \cdots \\ x_N^{(k-1)} \end{pmatrix} \right) ./ \begin{pmatrix} a_{1,1} \\ a_{2,2} \\ \cdots \\ a_{N,N} \end{pmatrix} \Rightarrow \mathbf{x}^{(k)} = (\mathbf{b} - (\mathbf{A} - diag(diag(\mathbf{A}))\, \mathbf{x}^{(k-1)})./diag(\mathbf{A})$$

```
%%% Example 12.3: Iterative solution
     x=[0 0 0]';                  % Initial estimate of the solution
     delta=0.0001; MAX=50; Z=x';  % Precission, iterations
     for i=1:MAX
       x=(b-(A-diag(diag(A)))*x)./diag(A);
       Z=[Z;x']; [m,n]=size(Z);
       if max(abs(Z(m,:)-Z(m-1,:)))<delta, break, end
     end; x, plot(Z)
```



EXAMPLES 12

12.1 Evaluate symbolic and numeric solution of a selected system of linear equations

12.2 Study methods of solution of sparse systems of linear equations

13. APPROXIMATION OF FUNCTIONS

Problem statement: Approximation of given values $\{x(i), y(i)\}_{i=1}^{N}$ by a function $f(\mathbf{a}, x)$ of parameters $\mathbf{a} = \{a(j)\}_{j=1}^{M}$

Solution:
1. The choice of the type of approximation function
   - Linear combination of base functions: $f(\mathbf{a}, x) = \sum_{j=1}^{M} a(j)\, g_j(x)$
   - Nonlinear function
2. Minimization of the mean square error:

$$S(\mathbf{a}) = \sum_{i=1}^{N} (f(\mathbf{a}, x(i)) - y(i))^2 \tag{1}$$

3. Evaluation of constants $\mathbf{a}$ of the approximation function $f(\mathbf{a}, x)$

COMMANDS

SUM
INV

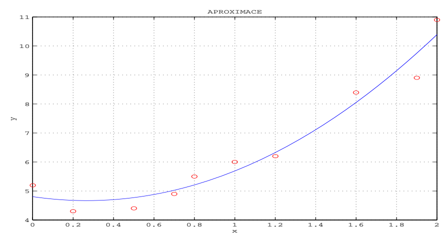
POLYFIT
POLYVAL

STEM
MESH
MESHC
CONTOUR

HOLD ON
HOLD OFF

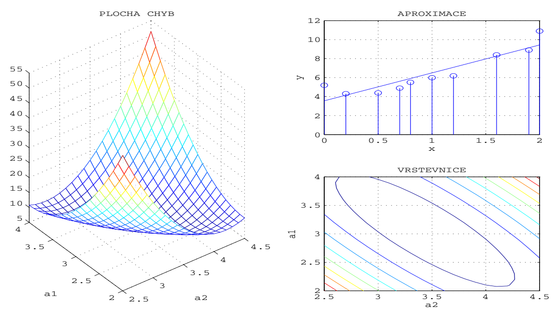
13.1 The Mean Square Error Method

```
%%% Example 13.1: Derive normal equations for approximation of values (x(i),y(i))
%%% for i=1,..,N by a function f(x)=a(1)*x^M+a(2)*x^(M-1)+...+a(M)*x+a(M+1), M=2
% Definition of given values
    x=[0 0.2 0.5 0.7 0.8 1 1.2 1.6 1.9 2]';
    y=[5.2 4.3 4.4 4.9 5.5 6 6.2 8.4 8.9 10.9]';
% Definition of the matrix of normal equations and its solution
    A=[sum(x.^4) sum(x.^3) sum(x.^2);...
       sum(x.^3) sum(x.^2) sum(x);...
       sum(x.^2) sum(x)    length(x)];
    b=[sum(x.^2.*y) sum(x.*y) sum(y)]'; a=inv(A)*b
% Plot of given and approximation values
    xx=[0:0.05:2]'; f=a(1)*xx.^2+a(2)*xx+a(3); plot(x,y,'or',xx,f);
    grid on; xlabel('x'); ylabel('y'); title('APPROXIMATION');
```



13.2 Library Functions

```
%%% Example 13.2: Approximate values (x(i),y(i)), for i=1,2,...,N by a
%%%    polynomial f(x)=a(1)*x^M+a(2)*x^(M-1)+ ... a(M)*x+a(M+1) for M=1
% Definition of given values
    x=[0 0.2 0.5 0.7 0.8 1 1.2 1.6 1.9 2]';
    y=[5.2 4.3 4.4 4.9 5.5 6 6.2 8.4 8.9 10.9]';
% Evaluation of coefficients a and values of
% the approximation function
    a=polyfit(x,y,1), xx=[0:0.05:2]'; f=polyval(a,xx);
% Evaluation of the sum of square errors for given ranges
% of parameters a1, a2 and plot of results
    a1=2:0.1:4; a2=2.5:0.1:4.5;
    for i=1:21; for j=1:21
        S(i,j)=sum((a1(j)*x+a2(i)-y).^2); end; end
    subplot(2,2,2); plot(xx,f); hold on; stem(x,y); hold off;
        xlabel('x'); ylabel('y'); title('APPROXIMATION'); grid on
    subplot(1,2,1); mesh(a2,a1,S); grid on; xlabel('a2'); ylabel('a1'); title('ERROR SURFACE');
    subplot(2,2,4); contour(a2,a1,S); xlabel('a2'); ylabel('a1'); title('CONTOUR PLOT');
```



EXAMPLES 13

13.1 Approximate the given sequence of values
  x=[0.3 0.4 0.6 0.9 1.5 2]';
  y=[11.7 10 8.3 7.2 6.3 6]';
by a function $f(x) = a(1)/x + a(2)$ and plot results

13.2 Approximate sequence of values
  x=[0 0.2 0.5 0.7 1.1 1.4 1.6 1.7 2]';
  y=[1 1.2 1.9 2.5 4.6 7.1 9.3 10.5 15]';
by polynomials of orders $M = 0, 1, ..., 5$ and plot dependence of the mean square error with respect to the polynomial order

14. LINEAR AND NONLINEAR APPROXIMATION

14.1 General Linear Approximation

```
%%% Example 14.1: Approximate values (x(i),y(i)), pro i=1,2,...,N
%%%    by a function f(x)=a(1)*x^3+a(2)*x
% Definition of given values
    x=[0 0.2 0.5 0.7 0.8 1 1.2 1.6 1.9 2]';
    y=[0 0.2 0.6 1.0 1.3 2 2.9 5.7 8.8 10]';
% Definition of the matrix of normal equations and its solution
    A=[x.^3 x]; a=A\y
% Plot of given and approximation values
    xx=0:0.05:2; f=a(1)*xx.^3+a(2)*xx; plot(x,y,'or',xx,f); grid on
      xlabel('x'); ylabel('y'); title('APPROXIMATION');
```

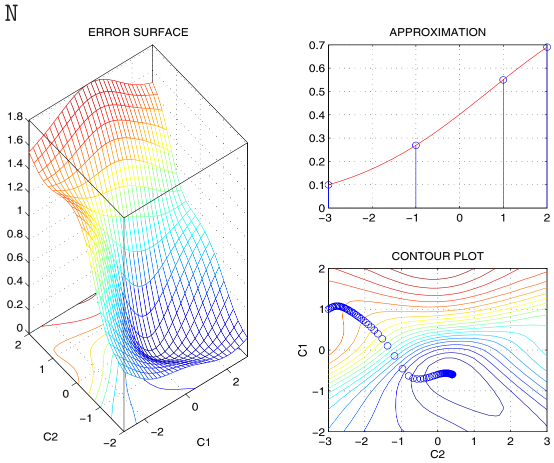
14.2 Nonlinear Approximation and Gradient Method

Problem statement: Approximation of given values $\{x(i), y(i)\}_{i=1}^{N}$ by a function $f(\mathbf{a}, x)$ of parameters $\mathbf{a} = \{a(j)\}_{j=1}^{M}$

Solution:

1. Statement of the mean square error evaluation:
$$S(\mathbf{a}) = \sum_{i=1}^{N} (f(\mathbf{a}, x(i)) - y(i))^2 \tag{2}$$
2. Coefficients estimation: coefficients $\mathbf{a}$, α and loop including
3. Gradient evaluation $DA1 = \partial(S)/\partial(a(1); DA2 = \partial(S)/\partial(a(2); ...$
4. Update of coefficients: $a(1) = a(1) - \alpha * DA1; a(2) = a(2) - \alpha * DA2; ...$

```
%%% Example 13.2: Approximate values (x(i),y(i)), pro i=1,2,...,N
%%%    by a function f(x)=f(x)=1/(exp(c1*x+c2)+1)
% Definition of given values and error surface plot
  x=[-3 -1 1 2]'; y=1./(exp(-0.6*x+0.4)+1);
  C1=-2:0.2:2; C2=-3:0.2:3;
  for i=1:length(C1); for j=1:length(C2)
    S(i,j)=sum((y-fa([C1(i),C2(j)],x)).^2); end; end
% Solution by gradient method
  c10(1)=1; c20(1)=-3; alpha=0.8; M=250; plot(c20,c10,'o');
  for k=1:M
    dc1=sum((y-fa([c10(k),c20(k)],x)).*...
      exp(c10(k)*x+c20(k))./(exp(c10(k)*x+c20(k))+1).^2 .*x);
    dc2=sum((y-fa([c10(k),c20(k)],x)).*...
      (exp(c10(k)*x+c20(k)))./(exp(c10(k)*x+c20(k))+1).^2);
    if(abs(dc1)+abs(dc2)<0.000001), break; end;
      c10(k+1)=c10(k)-alpha*dc1; c20(k+1)=c20(k)-alpha*dc2;end;
% Plot of given and approximation values
  subplot(1,2,1); meshc(C2,C1,S); axis tight
  subplot(2,2,4); contour(C2,C1,S,50);
    hold on; plot(c20,c10,'o'); hold off
  subplot(2,2,2); stem(x,y); hold on;
   xx=min(x):0.1:max(x); plot(xx,fa([c10(end), c20(end)],xx),'r'); hold off;
   grid on;
```




```
function f=fa(c,x)
  f=1./(exp(c(1)*x+c(2))+1);
```

COMMANDS

backslash

LSQLIN
LSQNONLIN
FMINSEARCH

LSQCURVEFIT

SOLVE

EXAMPLES 14

14.1 Approximate given sequence

x=[0.3 0.4 0.6 0.9 1.5 2]';
y=[0.4 0.6 1.0 1.7 3.8 6]';

by a function $f(x) = x + a(2) * x^2$ and plot results

15. STATISTICAL DATA ANALYSIS

15.1 Basic statistical characteristics

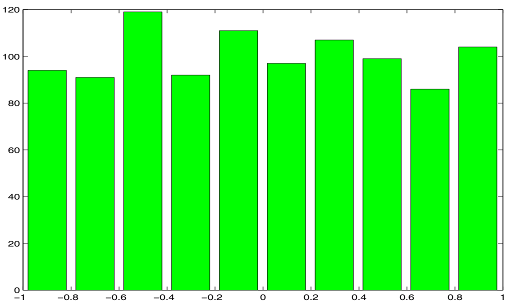
```
%%% Example 14.1: For the set of given values (x(i),y(i)), i=1,2,...,N evaluate
%%%    their mean value, standard deviation, and correlation coefficient

% Definition of given values
    x=[0 0.2 0.5 0.7 0.8 1 1.2 1.6 1.9 2]';
    y=x+0.1*rands(10,1);
% Evaluation
    [mean(x) mean(y)]
    [std(x) std(y)]
    corrcoef(x,y)
```

COMMANDS

MEAN
STD
CORRCOEF
RANDS

HIST
BAR

```
%%% Example 15.2: Evaluate and plot the histogram of distribution of
%%% random values rands(1000,1)

    [h,x]=hist(rands(1000,1));
    bar(x,h,'g')
```



EXAMPLES 15

15.1 Evaluate basic statistical characteristics of all columns of matrix $R = rands(100, 5)$

15.2 Analyse distribution of random values generated by function $R = randn(N, 1)$ for a chosen number N of its values

16. NONLINEAR EQUATIONS

Problem statement: Solution of equation: $f(x) = 0$

Symbolic solution: manipulation with expressions
Numeric solution: (i) separation of roots
(ii) iterative approximation of separate roots

16.1 Symbolic Solution

Characteristics:
1. Symbolic solution is not always possible
2. Substitution allows conversion to numerical solution

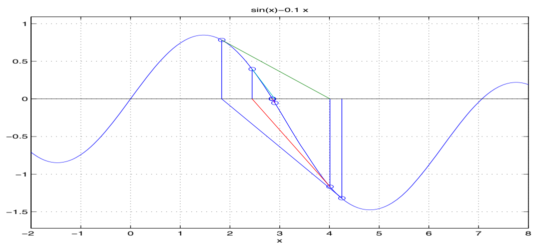
```
% Example 16.1: Symbolic solution of equation: ax^2+bx+c=0
  syms a b c x
  r=solve('a*x^2+b*x+c=0'); pretty(simple(r))
  r1=subs(r1,{a b c},{1 7 8})

% Example 16.2: Symbolic solution of equation:  tan(x)+sin(x)=2
  syms a b c x
  r=solve('tan(x)+sin(x)=2'), double(r)
  ezplot('tan(x)+sin(x)-2'); grid on
```

16.2 Numeric Solution

16.2.1 General Iterative Methods for Real Roots Estimation

Principle of the Newton Method:
1. Estimation of the initial approximation of the root: $x(1)$
2. Definition of the tangent to function $f(x)$ at point $[x(1), f(x(1))]$
$$y - f(x(1)) = f'(x)\ (x - x(1))$$
3. Intersection of the tangent with x-axis: $x(2) = x(1) - f(x(1))/f'(x(1))$

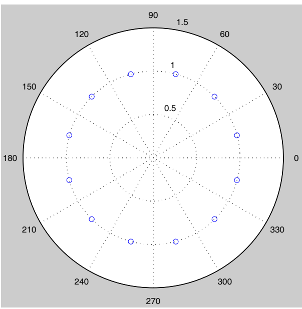
```
%%% Example 16.3: Solution of equation sin(x)-0.1x=0 by Newton method for
%%% initial approximation x(1), accuracy eps and maximum number of iterations M
    x(1)=4.25; eps=0.0001; M=10;
    for i=2:M
      x(i)=x(i-1)-f(x(i-1))/fd(x(i-1));
        if (abs(x(i)-x(i-1))<eps), break; end
    end
    ezplot('sin(x)-0.1*x',[-2 8]); grid on
      hold on; stem(x,f(x)); hold off
      line([x(1:end-1);x(2:end)],[f(x(1:end-1));...
                 zeros(1,length(x(2:end)))]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  function y=f(x)    function yd=fd(x)
  y=sin(x)-0.1*x;    yd=cos(x)-0.1;
```



16.2.2 Roots of Algebraic Equations

Equation definition: $f(x) \equiv c_1 x^n + c_2 x^{n-1} + \cdots + c_{n+1} = 0$

```
%%% Example 16.4: Solution of algebraic equation x^12+1=0
    c=[1 zeros(1,11) 1];
    r=roots(c)
      polar(angle(r), abs(r),'o'); grid on
    cc=poly(r)
```



COMMANDS

SYMS
SOLVE
SUBS
EZPLOT
PRETTY

ROOTS
POLY

POLAR

EXAMPLES 16

16.1 Evaluate symbolic and numeric solution of a selected nonlinear equation
16.2 Find and plot roots of selected algebraic equations

17. SYSTEM OF NONLINEAR EQUATIONS

Problem statement: Solution of equations: $f(x,y) = 0$
$g(x,y) = 0$

Symbolic solution: manipulation with expressions
Numeric solution: (i) separation of roots
(ii) iterative approximation of separate roots

COMMANDS

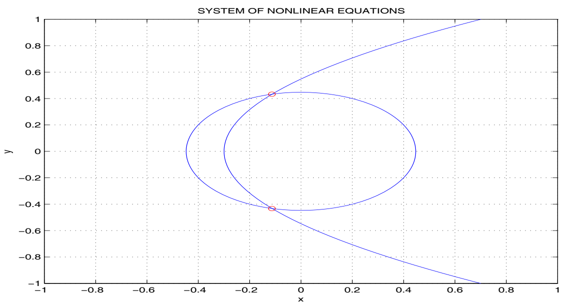
SYMS
SOLVE
DOUBLE
EZPLOT

FUNCTION

17.1 Symbolic Solution

Characteristics:
1. Symbolic solution is not always possible
2. Substitution allows conversion to numerical solution

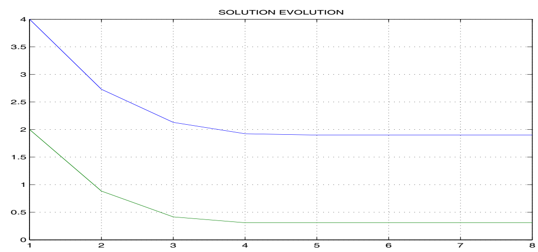
```
% Example 17.1: Symbolic solution of system of nonlinear equations
% Equation1: f(x,y)=x^2-2*x-y+0.5=0
% Equation2: g(x,y)=x^2+4*y^2-4=0
  syms x y
  R1=solve('x^2-2*x-y+0.5=0','x^2+4*y^2-4=0');
  R1.x; R1.y;
  X1=double(R1.x), Y1=double(R1.y)
% Visualization
  ezplot('x^2-2*x-y+0.5=0',[-1 3]); grid on
    hold on; ezplot('x^2+4*y^2-4=0');
      plot(X1([1 4]),Y1([1 4]),'or'); hold off
    title('SYSTEM OF NONLINEAR EQUATIONS')
```



17.2 Numeric Solution

Principle of the Newton method for the system of equations:
1. Expansion of multivariable functions into Taylor series is used
2. Principle is the same as in the one dimensional case

```
%%% Example 17.2: Solution of system of nonlinear equations
%%% Equation1: f(x,y)=x^2-2*x-y+0.5=0
%%% Equation2: g(x,y)=x^2+4*y^2-4=0
%%% for initial approximation P, accuracy eps and for
%%% maximum number of iterations M
    P=[4 2]'; eps=1e-12; M=40; PG=P;
    for k=2:M
      DF=J(P); F=[-f(P); -g(P)];
      DP=inv(DF)*F;
      P=P+DP; PG=[PG P];
      if abs(sum(DP))<eps, break, end
    end
    xk=P(:,end);
    plot(PG'); grid on; title('SOLUTION EVOLUTION')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  function z=f(P)           function z=g(P)
  x=P(1); y=P(2);           x=P(1); y=P(2);
  z=x^2-2*x-y+0.5;          z=x^2+4*y^2-4;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  function W=J(P)
  x=P(1); y=P(2);
  W=[(2*x-2) (-1);
     (2*x) (8*y)];
```



EXAMPLES 17

17.1 Evaluate symbolic solution of a selected system of nonlinear equations

17.2 Evaluate numeric solution of a selected system of nonlinear equations

18. NUMERICAL DIFFERENTIATION AND INTEGRATION

18.1 Interpolation

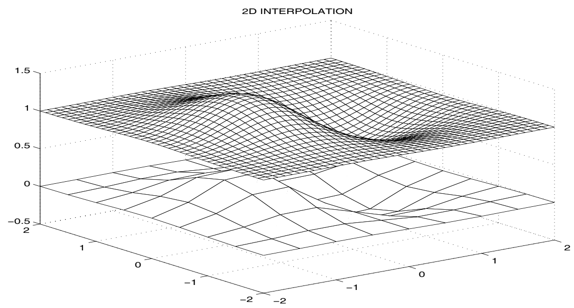
Characteristics:
1. Interpolation in one or more dimensions: linear, cubic, spline
2. Possibilities of triangulation based randomly located 3D data points

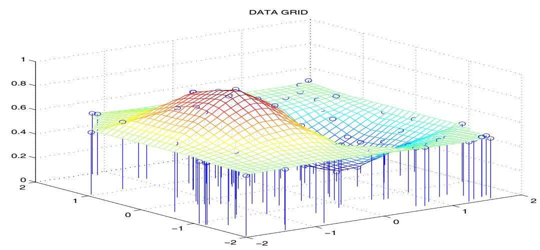
COMMANDS

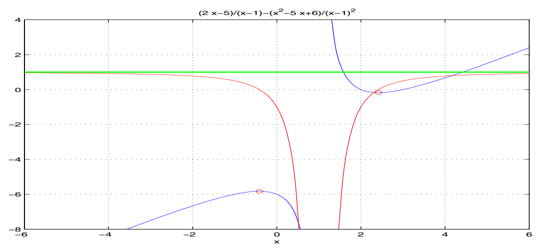
INTERP1
INTERP2
GRIDDATA

DIFF
LIMIT
INT
EZPLOT

```
% Example 18.1: 1D and 2D data interpolation
  figure(1); x=0:10; y=sin(x); plot(x,y,'o');
    xi=0:.2:10; yilin=interp1(x,y,xi); % 1D interpolation
                yisp=interp1(x,y,xi,'spline');
    hold on; plot(xi,yilin,'g',xi,yisp,'r'); hold off
  [X,Y]=meshgrid(-2:0.5:2); Z=-X.*exp(-X.^2-Y.^2);
  figure(2); mesh(X,Y,Z);
    [XI,YI]=meshgrid(-2:0.1:2);
    ZI=interp2(X,Y,Z,XI,YI,'spline'); % 1D interpolation
    hold on; mesh(XI,YI,ZI+1); hold off
% Example 18.2: data gridding
  x=4*(rand(100,1)-0.5); y=4*(rand(100,1)-0.5);
  z=-x.*exp(-x.^2-y.^2)+0.5;
    [XI,YI]=meshgrid(-2:0.1:2); ZI=griddata(x,y,z,XI,YI);
  figure(3); mesh(XI,YI,ZI); grid on;
    hold on; stem3(x,y,z,'o'); hold off
```







18.2 Numeric and Symbolic Derivative

Basic definitions:
1. Numerical estimate of a derivative: $y'(k) \approx (y(k+1) - y(k))/h$
2. Symbolic definition $f'(x) = \lim_{h\to 0}(f(x+h) - f(x))/h$

```
%%% Example 18.3: Numerical derivative
  a=0; b=pi; N=100; h=(b-a)/N;
  x=[a:h:b]; y=sin(x); yd=diff(y)/(pi/100);
%%% Example 18.4: Symbolic analysis of a given function
  syms x; f=(x^2-5*x+6)/(x-1); df=diff(f);
     pretty(f); pretty(df); pretty(simple(df))
  ezplot(f); hold on
    h=ezplot(df); set(h,'Color','r');
    hold off; axis([-6 6 -8 4]); grid on
% Evaluation and plot of the function derivative limits
  AP=limit(df,x,+inf); AM=limit(df,x,-inf);
  line([-6 6],double([AM AP]),'LineWidth',2,'Color','g')
% Evaluation and plot of extrems of the given function
  x0=double(solve(df)); extrem=[x0 subs(f,{x},{x0})]
  hold on; plot(extrem(:,1),extrem(:,2),'or'); hold off
```

18.3 Numeric and Symbolic Integration

Basic definitions:
1. Integration by a trapesoidal rule: $Q = \int_a^b f(x)\,dx \approx h/2 * (f(a) + 2\sum_{i=1}^{N-1} f(a+i\,h) + f(b))$

```
%%% Example 18.5: Numerical integration by the trapesoidal rule
  a=0; b=pi; N=100; h=(b-a)/N; x=[a:h:b]; y=sin(x);
  Q=h/2*(sin(a)+sin(b)+2*sum(sin(a+[1:N-1]*h)))
%%% Example 18.6: Symbolic integration
  syms x;  f=(x+1)/(x^2+5*x+6); ezplot(f); hold on
  fint=int(f); h=ezplot(fint); set(h,'Color','r');
     grid on; axis([-4 4 -4 4]); hold off
```

EXAMPLES 18

18.1 Using symbolical methods find extrems and limits of the given function

18.2 Derive estimates of the diference approximations the first and second derivative

18.3 Evaluate the definite integral of the given function using various numerical methods with a chosen step and compare results with the symbolical solution

19. NUMERICAL SOLUTION OF DIFFERENTIAL EQUATIONS

Problem statement: Solution of equation: $y' = f(x, y)$, $y(x_1) = y_1$

Solution: (i) analytical: function $y(x)$

(ii) numerical: sequence $\{x_i, y_i\}$, $i = 2, 3, \cdots$

COMMANDS

SYMS
DSOLVE
PRETTY
EZPLOT

ODE23
HOLD ON
HOLD OFF
PLOT

19.1 Symbolic Solution

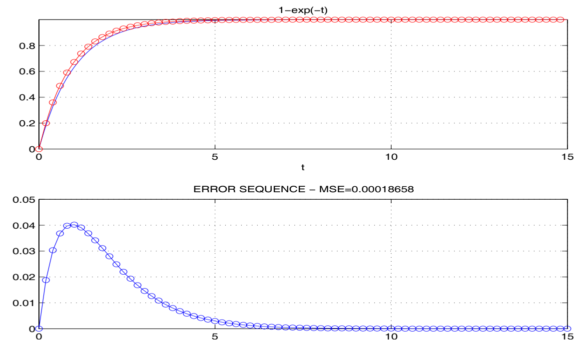
Characteristics:
1. Symbolic solution is not always possible
2. Substitution allows conversion to numerical solution

```
%%% Example 19.1: Ordinary Differential Equations - Initial-Value Problem
%%% Equation: y'+y=1, y(x1)=0; for x1=0 in range <x1, xk>
  clear; delete(get(0,'children')); syms t
  x1=0; xk=15;
  ys=dsolve('Dy+y=1','y(0)=0','t');
  subplot(2,1,1); ezplot(ys,[x1 xk]); grid on; axis tight
```

19.2 Numeric Solution

Principle of Euler method for the Initial Value Problem:
1. Approximate solution: $y_i = y_{i-1} + h\, f(x_{i-1}, y_{i-1})$, where $x_i = x_{i-1} + h$, $i = 2, 3, \cdots$ and h is a chosen step
2. Step value can affect accuracy and stability

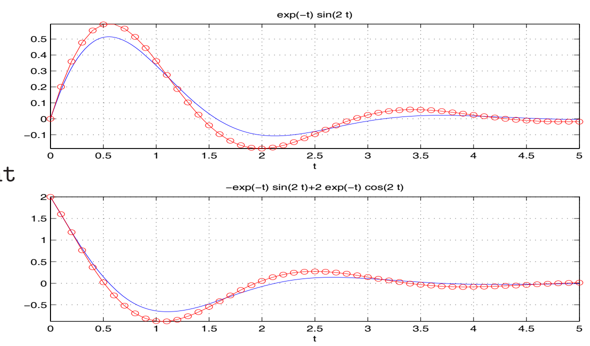
```
%%% Example 19.2: Ordinary Differential Equations - Initial-Value Problem
%%% Equation: y'=1-y, y(x1)=0; %%%%%%%%%%%%%% ... Cont of Ex.19.1 %%
  h=input('Step (=0.2): ');
  x(1)=x1; y(1)=0; N=(xk-x1)/h+1;
  for i=2:N
      x(i)=x(i-1)+h;
      y(i)=y(i-1)+h*(1-y(i-1)); end
  hold on; plot(x,y,'-or'); axis tight; hold off
  subplot(2,1,2); e=y-subs(ys,t,x); S=sumsqr(e)/length(e);
    plot(x,e,'-o');  grid on
    title(['ERROR SEQUENCE - MSE=',num2str(S)]);
```



19.3 Solution of System of Equations

Euler method application for solution of a system: $\mathbf{y}' = \mathbf{f}(x, \mathbf{y})$, $\mathbf{y}(x_1) = \mathbf{y}^{(1)}$
1. Vector solution: $\mathbf{y}^{(i)} = \mathbf{y}^{(i-1)} + h\,\mathbf{f}(x_{i-1}, \mathbf{y}^{(i-1)})$, where $x_i = x_{i-1} + h$, $i = 2, 3, \cdots$ and h is a chosen step
2. Step value can affect accuracy and stability

```
%%% Example 19.3: System of Ordinary Differential Equations - Initial-Value Problem
% y''+2y'+5y=0, y(x1)=0; y'(x1)=2 for x1=0 in range <x1, xk>
%%%%%% A. Symbolic Solution %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  clear; delete(get(0,'children')); syms t; x1=0; xk=5;
  ys=dsolve('D2y+2*Dy+5*y=0','y(0)=0','Dy(0)=2');
    pretty(simplify(ys)); ezplot(ys,[x1 xk]); axis tight
  subplot(2,1,1); ezplot(ys,[x1 xk]); grid on; axis tight
  subplot(2,1,2); ezplot(diff(ys),[x1 xk]); grid on; axis tight
%%%%%% B. Numeric Solution: Euler Method %%%%%%%%%%%%%%%
% Equation: y1'=-2*y1-5*y2; y1(x1)=2
%           y2'=y1;         y2(x1)=0;
  h=input('Step (=0.1): ');
  x(1)=x1; y1(1)=2; y2(1)=0; N=(xk-x1)/h+1;
  for i=2:N
      x(i)=x(i-1)+h; y1(i)=y1(i-1)+h*(-2*y1(i-1)-5*y2(i-1));
                     y2(i)=y2(i-1)+h*y1(i-1); end
  subplot(2,1,1); hold on; plot(x,y2,'-or'); axis tight; hold off
  subplot(2,1,2); hold on; plot(x,y1,'-or'); axis tight; hold off
```



EXAMPLES 19

19.1 Evaluate symbolic solution of a selected system of differential equations with given initial conditions

19.2 Evaluate numeric solution of a selected system of differential equations with given initial conditions

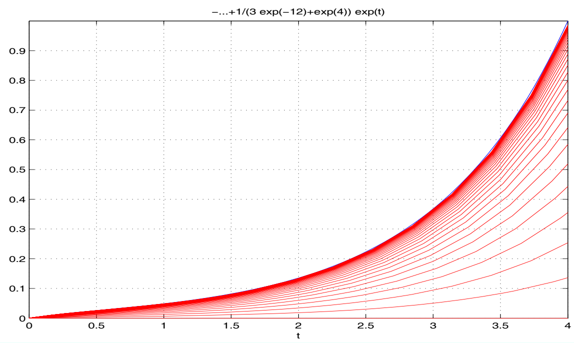
20. BOUNDARY PROBLEM

Solution of equation: $f(x, y, y', y'') = 0$, $y(x_a) = y_a$, $y(x_b) = y_b$

20.1 Shooting Method

Principle:
1. Estimate of the second (and further) initial conditions: $\widehat{y}'(x_a)$
2. Initial value problem solution to find $\hat{y}(x_b)$
3. The new estimate of the second initial condition:
   $$\widehat{y}'(x_a) = \widehat{y}'(x_a) - \alpha\,(\hat{y}(x_b) - y(x_b)) \text{ for a chosen } \alpha$$

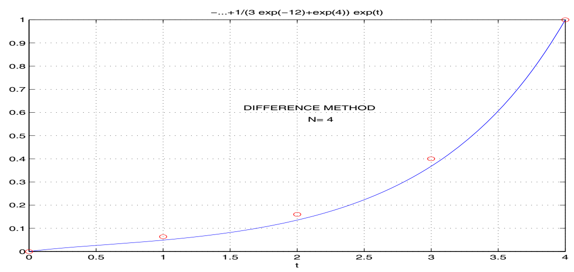
```
%%% Example 20.1: Ordinary Differential Equations - Boundary Problem
%%% Shooting Method: y''+2y'-3y=0, y(xa)=0; y(xb)=1 for xa=0, xb=4
%%%%%% A. Symbolic Solution %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  clear; delete(get(0,'children'))
  syms t; xa=0; xb=4;
  ys=dsolve('D2y+2*Dy-3*y=0','y(0)=0','Dy(4)=1');
    pretty(simplify(ys));
    ezplot(ys,[xa xb]); grid on; axis tight
%%%%%% B. Numeric Solution: Shooting Method %%%%%%%%%%%%%
% Equation: y1'=-2*y1+3*y2; y1(xa)=...
%           y2=y1;          y2(xa)=0;
  y1a=input('Volba pocatecni podminky (=0): ');
  y2a=0; y2b=1; alpha=0.01;
  for i=1:30
    [x,y]=ode23('f',[xa xb],[y1a y2a]);
     hold on; plot(x,y(:,2),'r'); axis tight; hold off
    y1a=y1a-alpha*(y(end,2)-y2b); end
```



20.2 Difference Method

Principle:
1. Division of the range $\langle x_a, x_b\rangle$ into N strips defining $x_k = xa + (k-1)\,h$ for $k = 2, 3, \cdots, N$ and $h = (xb - xa)/N$
2. Application of difference approximation of derivatives in equation $f(x_k, y_k, y'_k, y''_k) = 0$ for $k = 2, 3, \cdots, N$
3. Solution of the system of equations: $f(x_k, y_k, 1/h\,(y_{k+1} - yk - 1), 1/h^2\,(y_{k-1} - 2\,y_k + yk + 1)) = 0$
   for $k = 2, 3, \cdots, N$ where $y_1 = ya$ and $y_{N+1} = yb$

```
%%% Example 20.2: Ordinary Differential Equations - Boundary Problem
%%% Difference Method: y''+2y'-3y=0, y(xa)=0; y(xb)=1 for xa=0, xb=4
%%%%%% A. Symbolic Solution %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  clear; delete(get(0,'children'))
  syms t; xa=0; xb=4;
  ys=dsolve('D2y+2*Dy-3*y=0','y(0)=0','Dy(4)=1');
    pretty(simplify(ys));
    ezplot(ys,[xa xb]); grid on; axis tight
%%%%%% B. Numeric Solution: Difference Method %%%%%%%%%%%
% Equation: y''(k)+2*y(k)+3*y(k); y(xa)=0; y(xb)=1;
%     h=(xb-xa)/N;
%     (1/h^2)(y(k-1)-2y(k)+y(k+1)) + 2 (1/(2h))(y(k+1)-y(k-1)) - 3 y(k)=0
  N=4; h=(xb-xa)/N; x=xa:h:xb;
  ya=0; yb=1;
  A=[(-2/h^2-3)   (1/h^2+1/h)    0
     (1/h^2-1/h) (-2/h^2-3)  (1/h^2+1/h)
         0     (1/h^2-1/h) (-2/h^2-3)  ];
  b= [-(1/h^2-1/h)*ya  0  -(1/h^2+1/h)*yb]';  y=inv(A)*b; y=[ya; y; yb];
  hold on; t=plot(x,y,'or'); axis tight; hold off
```




```
 function yd=f(x,y)
% Function for Ex.20.1
  yd=[-2*y(1)+3*y(2); y(1)];
```

COMMANDS

SYMS
DSOLVE
PRETTY
EZPLOT

ODE23
INV
PLOT

EXAMPLES 20

20.1 Evaluate symbolic solution of a boundary value problem for ordinary differential equations using the shooting method

20.1 Evaluate symbolic solution of a boundary value problem for ordinary differential equations using the finite difference method

21. BOUNDARY PROBLEM SIMULATION

Principle:

1. Construction of a SIMULINK model for solution of a differential equation $f(x, y, y', y'') = 0$ for $y(x_a) = y_a$ and a chosen value of $\widehat{y'}(x_a)$ in the range $\langle x_a, x_b \rangle$
2. Construction of a MATLAB programme to use the SIMULINK model for the initial value problem to evaluate value $\hat{y}(x_b)$ and its use for estimation of the new value of the second initial condition:
$$\widehat{y'}(x_a) = \widehat{y'}(x_a) - \alpha\,(\hat{y}(x_b) - y(x_b)) \text{ for a chosen } \alpha$$
3. Iterative repetition

BLOCKS

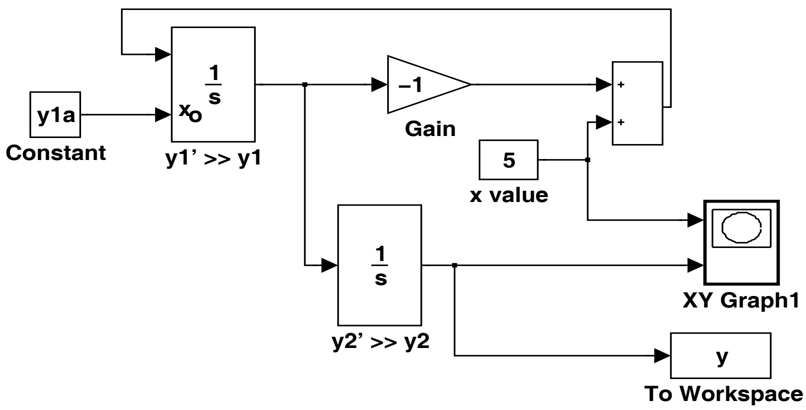
INTE-
GRATION

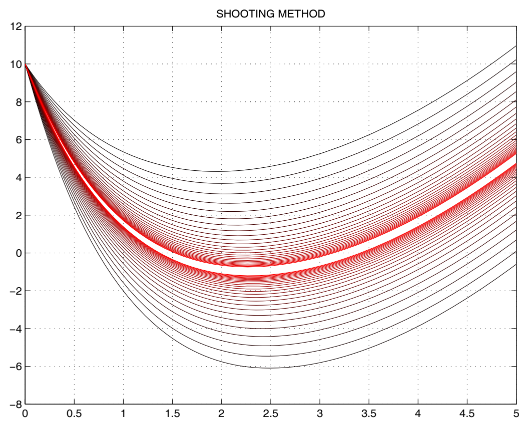
SUMMATION
CONSTANT
CLOCK

XYGRAPH

TO
WORKSPACE

```
%%% Example 21.1: % Solution of  the boundary problem by shooting method
%%% Using simulation in the SIMULINK environment
%%% Differential equation f(x,y,y',y'')=y''+y'-x=0, y(0)=10, y(5)=5
  clear all; close all; clc
  y1a=input('The choice on initial condition (=-20): ')
% Simulation
  alpha=1.95; y2b=5; M=50;
  BoundaryProblem; sim('BoundaryProblem')
  for i=1:M
    y1a=y1a-alpha*(y.signals.values(end,1)-y2b);
    sim('BoundaryProblem')
    plot(tout,y.signals.values,'Color',[1/M*i 0 0]); grid on; hold on
    pause(0.2)
  end
  hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  function dy=ff(x,y) dy=[-y(1)+x; y(1)];
```





Notes:

1. Depending upon the value of α the whole process can be stable or unstable, monotonic or oscilating
2. The SIMULINK run is controlled by the MATLAB programme

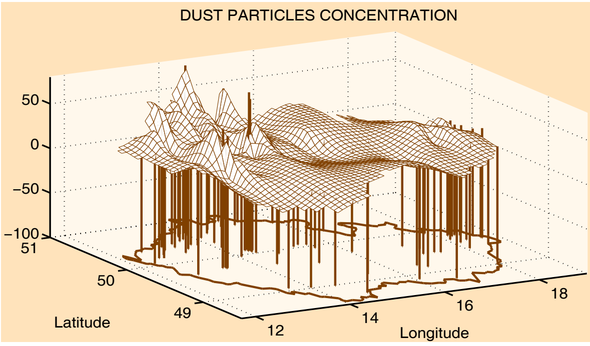
EXAMPLES 21

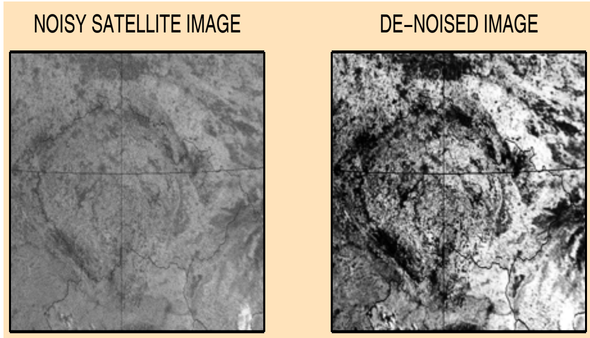
21.1 Evaluate solution of a boundary value problem for ordinary differential equations in the SIMULINK environment using the shooting method

21.2 Compare numeric solution obtained in the previous example with the symbolic one

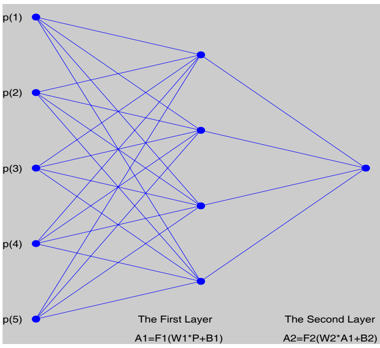
22. Applications

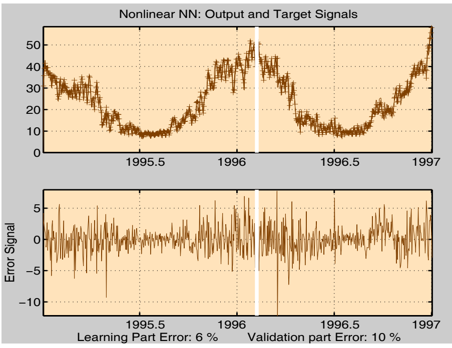
- Interpolation of air pollution data observed by ground measuring stations in specified locations inside the Czech Republic
- Satellite data of air pollution de-noising

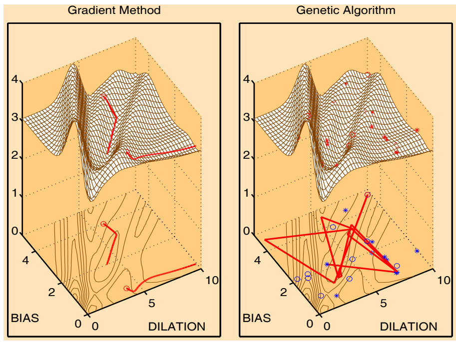




- Optimization of neural networks
- Prediction od gas consumption data







- Visualization of biomedical data
- Image enhancement
- Three-dimensional modelling