NUMERICAL ANALYSIS IN MATLAB

BASIC COMMANDS AND FUNCTIONS
OF THE VISUALIZATION 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
1. MATLAB Environment
2. Basic Operations
3. Control Commands
4. Subroutines
5. Objects and 2D Graphics
6. 3D Graphics
7. Data Files
8. MATLAB Graphical User Interface
9. Multidimensional Arrays
10. Symbolic Mathematics
11. SIMULINK Environment
12. Symbolic and Numeric Solution of Systems of Linear Equations
13. Approximation of Functions
14. Linear and Nonlinear Approximation
15. Statistical Data Analysis
16. Symbolic and Numeric Solution of Nonlinear Equations
17. Systems of Nonlinear Equations
18. Symbolic and Numeric Differentiation and Integration
19. Symbolic and Numeric Solution of Differential Equations
20. Boundary Problem of the Solution of Differential Equations
21. Boundary Problem Simulation
22. Applications
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 \( \Rightarrow \) store of a variable without its visualization
2. Return to previously written commands \( \Rightarrow \) up-arrow key
3. Help (help, help \( \langle \text{command} \rangle \), lookfor \( \langle \text{key} \rangle \), ...) and DOS commands (dir,...)
4. Visualization of variables (whos), delete of variables (clear \( \langle \text{list} \rangle \))
5. The use of standard variables: \( \pi \), \( \epsilon \), \( i \), \( j \)

%%% 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 \( x=a,a+h,..,b \)
```
>> semiprog % Programme call
```

**Solution of Examples**

\[
A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 6 \\ 15 \\ 16 \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}
\]

### EXAMPLES 1

1.1 Evaluate and plot function \( f(x) = \exp(x) \) for \( x \in [a, b] \) and step \( h \) \( (a=0, b=2, h=0.2) \)
1.2 Evaluate and plot function \( f(x) = \log(x) \) for \( x \in [a, b] \) 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: +, -, *, /, ^, ' (element-by-element operations)
- vector: *, ./, .^, .' (element-by-element operations)
- relational: >, >=, ==, ~=, <, <=
- logical: & | ~

2.1 Assignment Commands

\[(\text{variable}) = (\text{expression})\]

%% Example 2.1: Operations with vectors
\[
\begin{align*}
&v = [1 2 3 4]; \\
&r1 = v*v'; \\
&r2 = v.*v; \\
&r3 = v'*\text{ones}(1,4)
\end{align*}
\]

%% Example 2.2: Relational expressions
\[
\begin{align*}
&s1 = 3 > 5; \\
&s2 = 3 < 5
\end{align*}
\]

%% Example 2.3: Matrix operations
\[
\begin{align*}
&A = [1 2; 3 4], \\
&b = [1 2] \\
&C = [A, b'; [1 1 1]]
\end{align*}
\]

2.2 Functions

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

%% Example 2.4: Functions
\[
\begin{align*}
&A = [1 2; 4 5]; \\
&f1 = \text{sin}(A) \%\text{ scalar function} \\
&f2 = \text{sum}(A) \%\text{ vector function} \\
&f3 = \text{det}(A) \%\text{ matrix function}
\end{align*}
\]

%% Example 2.5: Special functions
\[
\begin{align*}
&z1 = \text{zeros}(2,3) \\
&z2 = \text{ones}(1,3) \\
&z3 = \text{rand}(4,3) \\
&\text{plot}(z3); \text{title('PLOT OF MATRIX R=RAND(4,3)')}
\end{align*}
\]

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 \(v=\text{RAND}(1,N)\) 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

```plaintext
for ⟨variable⟩ = ⟨expression⟩
  ⟨commands⟩
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:
%%% \[ Q \approx h \sum_{i=1}^{N} f(x(i)) \]
%%% \( x = a + (i-1)h, \ i = 1, 2, \ldots, N \)
```plaintext
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

```plaintext
if ⟨relational_expression_1⟩
  ⟨commands_1⟩
elseif ⟨relational_expression_2⟩
  ⟨commands_2⟩
else
  ⟨commands_3⟩
end
```

%%% Example 3.2: Plot of the given function in selected limits
%%% \( a=0, b=\pi \) for \( N=20, 40 \) and \( 60 \) its parts using the trapezoidal rule
```plaintext
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
```

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(:)'

4.3 Function Subroutines

function [⟨output_parameters⟩]=⟨name⟩(⟨input_parameters⟩)
⟨commands⟩

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)

EXAMPLES 4

4.1 Define function subroutine for evaluation of minimum and maximum value
of the given matrix and apply it for matrix \( A = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ 1 & 1 & 1 \end{bmatrix} \)

4.2 Modify the given vector \( x=[1.1 \ 1.4 \ 1.9 \ 2.1 \ 1.3 \ 1.7 \ 1.8 \ 5 \ 1.5] \) 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

\[
\langle \text{name} \rangle \langle \text{= \langle \text{command} \rangle ( \langle \text{property\_name} \rangle, \langle \text{property\_value} \rangle) \rangle
\]

%%% 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]);

5.2 Two-dimensional Graphics

%%% Example 5.2: Plot functions \( f_1(x) = \sin(x) \) and \( f_2(x) = \abs{\cos(x)} \) 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 \( x \in (a,b) \) 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 Fundamentals 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 \( X, Y \)
   \( \rightarrow \) function MESHGRID
2. Selection of 3D plotting \( \rightarrow \) function MESH, MESHC, ...
3. Selection of the viewpoint \( \rightarrow \) function VIEW
4. The choice of the number of contour lines and their description \( \rightarrow \) function CONTOUR

%%% Example 6.1: Plot function
%%% \( f(x,y) = -x \exp(-x^2-y^2) \) for \( x \in (-2,2) \) and \( y \in (-2,2) \)
%%% together with corresponding contour lines

\[
[X,Y]=\text{meshgrid}(-2:0.2:2);
Z=-X.*\exp(-X.^2-Y.^2);
\text{meshc}(X,Y,Z);
title('f(x,y)=-x*exp(-x^2-y^2)'); xlabel('x'); ylabel('y');
colormap([0 0 1])

6.2 Colors
Notes to the use of basic commands:

1. Palette of \( m \) colors is defined by values of matrix \( M_{m,3} \)
2. Each of \( m \) colors is defined by the combination of 3 basic colors (Red-Green-Blue) and their intensity in the range of \( (0,1) \)
3. Application of the color map \( \rightarrow \) function COLORMAP
4. Plot of the color map \( \rightarrow \) 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=\text{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 \( x \in (-2, 2) \) and \( y \in (-2.5, 2.5) \)

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

6.3 Visualize contour plot of function \( f(x, y) = -x^2y^2 \)
for \( x \in (-2, 2) \) and \( y \in (-2, 2) \) with line description
7. DATA FILES MANIPULATION

7.1 Data Saving

```
save (file_name) [(list_of_variables)] [-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

```matlab
A=[1 2 3; 4 5 6; 7 8 1]; b=[6 15 16]'; save dat1 A b;
```

7.2 Data Retrieving

```
load (file_name)
```

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

```matlab
load dat1
```

7.3 Data Import from EXCEL

Steps:
1. Opening of the MATLAB environment and EXCEL `filename.xls`
2. Data transport from EXCEL to MATLAB:
   ```matlab
chan=ddeinit('excel', 'filename.xls');
D1=ddereq(chan,'r1c1:r2c2');
```
   where r1, r2, c1, c2 stand for initial and final number of rows and columns

Example 7.3: Reading of the EXCEL table

```matlab
chan=ddeinit('excel', 'g2010.xls');
D1=ddereq(chan,'r10c3:r1000c3'); plot(D1)
```

COMMANDS

SAVE
LOAD
WHOS
DIR
DDEINT
DDEREQ

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

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

8.1 Steps of GUI Construction

1. Opening of the environment ➞ 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 ➞ 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

EXAMPLES 8

8.1 Modification of slider limit values
8.2 Modification of plotted function
9. MULTIDIMENSIONAL ARRAYS

9.1 Multidimensional Arrays

\[
\begin{align*}
& \text{\texttt{a(1,1,1)} ... \texttt{a(1,N,1)}} \\
& \vdots \\
& \text{\texttt{a(M,1,1)} ... \texttt{a(M,N,1)}} \\
& \text{\texttt{a(1,1,2)} ... \texttt{a(1,N,2)}} \\
& \vdots \\
& \text{\texttt{\ldots}} \text{\texttt{a(M,N,2)}} \\
& \text{\texttt{a(1,1,k)} ... \texttt{a(1,N,k)}} \\
& \vdots \\
& \text{\texttt{\ldots}} \text{\texttt{a(M,N,k)}}
\end{align*}
\]

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
\[
\begin{align*}
& \texttt{>> A(2,2,1)=1;} \\
& \texttt{>> A(2,2,2)=2;} \\
& \texttt{>> A} & \text{\% Variable display}
\end{align*}
\]

9.2 Structured Arrays

\[
\begin{align*}
\text{\texttt{STATION}} \\
& \text{\texttt{.NAME}} \\
& \text{\texttt{.POSITION}} \\
& \text{\texttt{.DATA}} \\
& \texttt{\{1 2 3\}} \\
& \texttt{10 11} \\
& \texttt{12 13}
\end{align*}
\]

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

%%% Example 9.2: Structured array definition
\[
\begin{align*}
& \texttt{>> STATION(1).NAME='SANTINKA';} \\
& \texttt{>> STATION(1).POSITION=[1 2 3];} \\
& \texttt{>> STATION(1).DATA=[10 11;12 13];} \\
& \texttt{>> STATION} & \text{\% Variable display} \\
& \texttt{>> D=STATION(1).DATA} & \text{\% Data selection}
\end{align*}
\]

9.3 Cells

\[
\begin{align*}
& \text{\texttt{CELL 1,1}} \\
& \begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{array} \\
& \text{\texttt{CELL 1,2}} \\
& \begin{array}{c}
'Jonathan Smith'
\end{array} \\
& \text{\texttt{CELL 2,1}} \\
& \text{\texttt{STATION}} \\
& \text{\texttt{STATION.NAME}} \\
& \text{\texttt{STATION.POSITION}} \\
& \text{\texttt{STATION.DATA}} \\
& \begin{array}{c}
1 2 3 4 5
\end{array} \\
& \text{\texttt{CELL 2,2}} \\
& \text{\texttt{\{1 2 3 4 5\}}} \\
& \text{\texttt{\% Variable display}} \\
& \text{\texttt{P=C(2,1).POSITION(2) \% Data selection}}
\end{align*}
\]

1. Data items can be of any kind

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

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 \(\Rightarrow\) 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: \(y'=\text{int}(-y)\) dy
3. Description of another integration block: \(y=\text{int}(y')\) dy

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
\[
\begin{pmatrix}
ad_{1,1} & ad_{1,2} & \cdots & ad_{1,N} \\
ad_{2,1} & ad_{2,2} & \cdots & ad_{2,N} \\
ad_{N,1} & ad_{N,2} & \cdots & ad_{N,N}
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
\cdots \\
x_N
\end{pmatrix}
= 
\begin{pmatrix}
b_1 \\
b_2 \\
\cdots \\
b_N
\end{pmatrix}
\iff
A_{N,N} x_{N,1} = b_{N,1}
\]

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; 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

12.2 Numerical Solution

12.2.1 Finite Methods
Mathematical background of Gauss-Jordan method:
\[
\begin{pmatrix}
ad_{1,1} & ad_{1,2} & \cdots & ad_{1,N} \\
ad_{2,1} & ad_{2,2} & \cdots & ad_{2,N} \\
ad_{N,1} & ad_{N,2} & \cdots & ad_{N,N}
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
\cdots \\
x_N
\end{pmatrix}
= 
\begin{pmatrix}
b_1 \\
b_2 \\
\cdots \\
b_N
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
10 & \cdots & 0 \\
01 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
00 & \cdots & 1
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
\cdots \\
x_N
\end{pmatrix}
\]

%%% 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:
\[
\begin{pmatrix}
xx(k) \\
x1(k) \\
\vdots \\
x_N(k)
\end{pmatrix}
= 
\begin{pmatrix}
b_1 \\
b_2 \\
\vdots \\
b_N
\end{pmatrix}
- 
\begin{pmatrix}
0 & a_{1,2} & \cdots & a_{1,N} \\
0 & a_{2,2} & \cdots & a_{2,N} \\
\vdots & \ddots & \vdots & \vdots \\
0 & a_{N,2} & \cdots & a_{N,N}
\end{pmatrix}
\begin{pmatrix}
xx(k-1) \\
x1(k-1) \\
\vdots \\
x_N(k-1)
\end{pmatrix}
\Rightarrow
xx(k) = (b - (A - diag(diag(A))) xx(k-1))./diag(A)
\]

%%% Example 12.3: Iterative solution
x=[0 0 0]'; % Initial estimate of the solution
delta=0.0001; MAX=50; Z=x'; % Precision, 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)

COMMANDS

SYMS
SOLVE
SUBS
SIMPLE
PRETTY
INV
DIAG
SPARSE
FULL
SPY
GPLOT

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 \( f(a, x) \) of parameters \( a = \{a(j)\}_{j=1}^{M} \)

Solution:
1. The choice of the type of approximation function
   - Linear combination of base functions: \( f(a, x) = \sum_{j=1}^{M} a(j) g_j(x) \)
   - Nonlinear function
2. Minimization of the mean square error:
   \[
   S(a) = \sum_{i=1}^{N} (f(a, x(i)) - y(i))^2
   \]  
3. Evaluation of constants \( a \) of the approximation function

13.1 The Mean Square Error Method

%%% Example 13.1: Derive normal equations for approximation of values \((x(i), y(i))\) for \(i = 1, \ldots, N\) by a function \( f(x) = a(1)*x^2 + a(2)*x + a(3) \) for \( M = 2 \)

% 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 \(i = 1, 2, \ldots, N\) by a polynomial \( f(x) = a(1)*x^2 + a(2)*x + a(3) \) for \( M = 1 \)

% Definition of given 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]';

% 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(2,2,1); mesh(a2,a1,S); grid on; xlabel('a2'); ylabel('a1'); title('ERROR SURFACE');
subplot(2,2,2); contour(a2,a1,S); xlabel('a2'); ylabel('a1'); title('CONTOUR PLOT');

EXAMPLES 13

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

13.2 Approximate sequence of values
\[
\begin{align*}
x &= [0 0.2 0.5 0.7 1.1 1.4 1.6 1.7 2]'; \\
y &= [1 1.2 1.9 2.5 4.6 7.1 9.3 10.5 15]';
\end{align*}
\] by polynomials of orders \( M = 0, 1, \ldots, 5 \) 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 \(i=1,2,\ldots,N\) 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

\[
A = [x.*x x]; a = A\backslash y
\]

Plot of given and approximation values

\[
x = 0:0.05:2; f = a(1)*x.^3 + a(2)*x; plot(x,y,'or',xx,f); grid on
\]

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 \(f(a,x)\) of parameters \(a = \{a(j)\}_{j=1}^{M}\)

Solution:

1. Statement of the mean square error evaluation:

\[
S(a) = \sum_{i=1}^{N} (f(a,x(i)) - y(i))^2
\]

2. Coefficients estimation: coefficients \(a, \alpha\) and loop including

3. Gradient evaluation

\[
DA_1 = \frac{\partial S}{\partial a(1)}; DA_2 = \frac{\partial S}{\partial a(2)}; \ldots
\]

4. Update of coefficients:

\[
a(1) = a(1) - \alpha * DA_1; a(2) = a(2) - \alpha * DA_2; \ldots
\]

Example 13.2: Approximate values \((x(i),y(i))\), pro \(i=1,2,\ldots,N\) by a function \(f(x)=\frac{1}{\exp(c1*x+c2)+1}\)

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 \(i=1:length(C1); for j=1:length(C2)\)

\[
S(i,j) = \text{sum}((y-fa([C1(i),C2(j)],x)).^2); end;
\]

Solution by gradient method

\[
c10(1)=1; c20(1)=-3; alpha=0.8; M=250; plot(c20,c10,'o');
\]

for \(k=1:M\)

\[
dc1 = \text{sum}((y-fa([c10(k),c20(k)],x)).*...\]

\[
\text{exp(c10(k)*x+c20(k)))./(exp(c10(k)*x+c20(k))+1)^2.*x);}
\]

\[
dc2 = \text{sum}((y-fa([c10(k),c20(k)],x)).*...\]

\[
\text{exp(c10(k)*x+c20(k)))./(exp(c10(k)*x+c20(k))+1)^2);}
\]

\[
\text{if(abs(dc1)+abs(dc2)<0.0000001), break; end;}
\]

\[
c10(k+1)=c10(k)-alpha*dc1; c20(k+1)=c20(k)-alpha*dc2;end;
\]

Plot of given and approximation values

\[
\text{subplot}(1,2,1); meshc(C2,C1,S); axis tight
\]

\[
\text{subplot}(2,2,4); contour(C2,C1,S,50);
\]

hold on; plot(c20,c10,'o'); hold off

\[
\text{subplot}(2,2,2); stem(x,y); hold on;
\]

\[
xx = \text{min}(x):0.1:\text{max}(x); plot(xx,fa([c10(end), c20(end)],xx),'r'); hold off; grid on;
\]

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
15.1 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)

%%% 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: \( ax^2+bx+c=0 \)
```matlab
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: \( \tan(x)+\sin(x)=2 \)
```matlab
syms abcx
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 \( \varepsilon \) and maximum number of iterations \( M \)
```matlab
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)))]);
```

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

16.2.2 Roots of Algebraic Equations

Equation definition: \( f(x) \equiv c_1x^n + c_2x^{n-1} + \cdots + c_{n+1} = 0 \)

%%% Example 16.4: Solution of algebraic equation \( x^{-12}+1=0 \)
```matlab
c=[1 zeros(1,11) 1];
r=roots(c);
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

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
Equation 1: \[ f(x, y) = x^2 - 2x - y + 0.5 = 0 \]
Equation 2: \[ g(x, y) = x^2 + 4y^2 - 4 = 0 \]

% Example 17.1: Symbolic solution of system of nonlinear equations
% Equation 1: f(x, y) = x^2 - 2x - y + 0.5 = 0
% Equation 2: g(x, y) = x^2 + 4y^2 - 4 = 0
syms x y
R1 = solve('x^2 - 2x - 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 - 2x - 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
Equation 1: \[ f(x, y) = x^2 - 2x - y + 0.5 = 0 \]
Equation 2: \[ g(x, y) = x^2 + 4y^2 - 4 = 0 \]

for initial approximation \( P \), accuracy \( \varepsilon \) and for
maximum number of iterations \( M \)

\[ P = [4 \ 2] ; \ \varepsilon = 1e-12 ; \ M = 40 ; \ PG = P ; \]
for \( k = 2:M \)
\[ DF = J(P) ; \ F = [-f(P) ; -g(P)] ; \]
\[ DP = \text{inv}(DF) * F ; \]
\[ P = P + DP ; \ PG = [PG P] ; \]
if abs(sum(DP)) < \( \varepsilon \), break, end
end
\[ x_k = P(:, end) ; \]
plot(PG'); grid on; title('SOLUTION EVOLUTION')

% Example 17.2: Solution of system of nonlinear equations
% Equation 1: f(x, y) = x^2 - 2x - y + 0.5 = 0
% Equation 2: g(x, y) = x^2 + 4y^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)
    x=Px(1); y=Px(2);
    z=x^2-2*x-y+0.5;
function z=g(P)
    x=Py(1); y=Py(2);
    z=x^2+4*y^2-4;

function W=J(P)
    x=Px(1); y=Py(2);
    W=[(2*x-2) (-1)
       (2*x) (8*y)];

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

% 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=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 \( f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \)

%%% 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');
grid on; axis([-6 6 -8 4]); hold off

18.3 Numeric and Symbolic Integration

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

%%% Example 18.5: Numerical integration by the trapezoidal rule
a=0; b=pi; 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;
fig=int(f);
h=ezplot(fig); 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 difference 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 \( \{x_i, y_i\}, i = 2, 3, \ldots \)

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: \( y' + y = 1, y(x_1) = 0 \); \( x_1 = 0 \) in range \( <x_1, x_0> \)
clear; delete(get(0,'children')); syms t
x1=0; xk=15;
y=dsolve('Dy+y=1','y(0)=0','t'); subplot(2,1,1); ezplot(y,[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 \( x_i = x_{i-1} + h, \) \( i = 2, 3, \ldots \)
   and \( h \) is a chosen step
2. Step value can affect accuracy and stability

%%% Example 19.2: Ordinary Differential Equations - Initial-Value Problem
%%% Equation: \( y' = 1 - y, y(x_1) = 0 \); \( x_1 = 0 \) in range \( <x_1, x_0> \)
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(y,'t,x'); S=sumsqr(e)/length(e);
plot(x,e,'-o'); grid on

19.3 Solution of System of Equations

Euler method application for solution of a system: \( y' = f(x, y), y(x_1) = y_1 \)
1. Vector solution:
   \( y_i = y_{i-1} + h f(x_{i-1}, y_{i-1}), \) where \( x_i = x_{i-1} + h, \) \( i = 2, 3, \ldots \)
   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
%%% A. Symbolic Solution
clear; delete(get(0,'children')); syms t; x1=0; xk=5;
y=dsolve('D2y+2*Dy+5*y=0','y(0)=0','Dy(0)=2'); pretty(simplify(y)); ezplot(y,[x1 xk]); axis tight
subplot(2,1,1); ezplot(y,[x1 xk]); grid on; axis tight
subplot(2,1,2); ezplot(diff(y),[x1 xk]); grid on; axis tight
%%% B. Numeric Solution: Euler Method
% y''+2y'+5y=0, y(x1)=0; y'(x1)=2 for x1=0 in range <x1, xk>
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

COMMANDS
SYMS
DSOLVE
PRETTY
EZPLOT
ODE23
HOLD ON
HOLD OFF
PLOT

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: \( \hat{y}'(x_a) \)
2. Initial value problem solution to find \( \hat{y}(x_b) \)
3. The new estimate of the second initial condition:
\[
\hat{y}'(x_a) = \hat{y}'(x_a) - \alpha (\hat{y}(x_b) - y(x_b)) \text{ for a chosen } \alpha
\]

%%% 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;
y=dsolve('D2y+2*Dy-3*y=0','y(0)=0','Dy(4)=1');
pretty(simplify(y));
ezplot(y,[xa xb]); grid on; axis tight

%%% B. Numeric Solution: Shooting Method %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Equation: \( y'(x_n)+2y(x_n)+3y(x_n); \ y(xa)=0; \ y(xb)=1; \ h=(xb-xa)/N; \)
% \( (1/h^2)(y(k-1)-2y(k)+y(k+1)) + 2 (1/(2h))(y(k+1)-y(k-1)) - 3 y(k)=0 \)
% for \( k=2,3,...,N \) where \( y_1=ya \) and \( y_{N+1}=yb \)

%%% Example 20.2: Ordinary Differential Equations - Boundary Problem
%%% Difference Method: \( y''(k)+2y(k)+3y(k); \ y(xa)=0; \ y(xb)=1; \)
% \( h=(xb-xa)/N; \)
% \( [y(1/h^-2)](y(k-1)-2y(k)+y(k+1)) + 2 [1/(2h)](y(k+1)-y(k-1)) - 3 y(k)=0 \)
% \( N=4; \ h=(xb-xa)/N; \ x=x:%a:h:xb; \)
y=x;x=xb;2=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 2y(2)+3y(2); y=inv(A)*b; y=[ya; y; yb];
hold on; t=plot(x,y,'or'); axis tight; hold off

20.2 Difference Method

Principle:
1. Division of the range \( (x_a, x_b) \) into \( N \) strips defining \( x_k = x_a + (k-1) h \) for \( k=2,3,...,N \) and \( h=(x_b-x_a)/N \)
2. Application of difference approximation of derivatives in equation \( f(x_k, y_k, y'_k, y''_k) = 0 \) for \( k=2,3,...,N \)
3. Solution of the system of equations:
\[
 f(x_k, y_k, 1/h (y_{k+1} - y_{k-1}), 1/h^2 (y_{k+1} - 2 y_k + y_{k+1})) = 0
\]
for \( k=2,3,...,N \) 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;
y=dsolve('D2y+2*Dy-3*y=0','y(0)=0','Dy(4)=1');
pretty(simplify(y));
ezplot(y,[xa xb]); grid on; axis tight

%%% B. Numeric Solution: Difference Method %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Equation: \( y''(k)+2y(k)+3y(k); \ y(xa)=0; \ y(xb)=1; \)
% \( h=(xb-xa)/N; \)
% \( [y(1/h^-2)](y(k-1)-2y(k)+y(k+1)) + 2 [1/(2h)](y(k+1)-y(k-1)) - 3 y(k)=0 \)
% \( N=4; \ h=(xb-xa)/N; \ x=x:%a:h:xb; \)
y=x;x=xb;2=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 2y(2)+3y(2); y=inv(A)*b; y=[ya; y; yb];
hold on; t=plot(x,y,'or'); axis tight; hold off
21. BOUNDARY PROBLEM SIMULATION

Principle:
1. Construction of a SIMULINK model for solution of a differential equation
   \[ f(x, y, y', y'') = 0 \] for \( y(x_0) = y_a \) and a chosen value of \( \hat{y}'(x_a) \) in the range \((x_a, x_b)\)
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:
   \[ \hat{y}'(x_a) = \hat{y}'(x_a) - \alpha (\hat{y}(x_b) - y(x_b)) \] for a chosen \( \alpha \)
3. Iterative repetition

%%% 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 \( \alpha \) the whole process can be stable or unstable, monotonic or oscillating
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 of gas consumption data

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