

Scilab Textbook Companion for  
Introductory Methods Of Numerical Analysis  
by S. S. Sastry<sup>1</sup>

Created by  
Rohit Kumar Singh  
B Tech  
Others  
NIT, JAMSHEDPUR  
College Teacher  
NA

Cross-Checked by  
Lavitha Pereira

May 24, 2016

<sup>1</sup>Funded by a grant from the National Mission on Education through ICT, <http://spoken-tutorial.org/NMEICT-Intro>. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website <http://scilab.in>

# Book Description

**Title:** Introductory Methods Of Numerical Analysis

**Author:** S. S. Sastry

**Publisher:** Phi Learning

**Edition:** 5

**Year:** 2012

**ISBN:** 9788120345928

Scilab numbering policy used in this document and the relation to the above book.

**Exa** Example (Solved example)

**Eqn** Equation (Particular equation of the above book)

**AP** Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

# Contents

List of Scilab Codes	4
1 Errors in numerical calculation	5
2 Solution of Algebraic and Transcendental Equation	13
3 interpolation	40
4 least squares and fourier transform	61
5 spline functions	77
6 Numerical Diffrentiation and Integration	83
7 Numerical linear algebra	103
8 Numerical Solution of ordinary differential equation	117
9 Numerical Solution of Partial Differential Equation	132

# List of Scilab Codes

Exa 1.1	rounding off . . . . .	5
Exa 1.2	percentage accuracy . . . . .	5
Exa 1.3	absolute and relative errors . . . . .	6
Exa 1.4	best approximation . . . . .	6
Exa 1.5	relative error . . . . .	7
Exa 1.6	absolute error and relative error . . . . .	8
Exa 1.7	absolute error . . . . .	8
Exa 1.8	difference in 3 significant figures . . . . .	9
Exa 1.10	relative error . . . . .	9
Exa 1.11	relative error . . . . .	9
Exa 1.12	taylor series . . . . .	10
Exa 1.13	taylor series . . . . .	10
Exa 1.14	maclaurins expansion . . . . .	11
Exa 1.15	series approximation . . . . .	12
Exa 2.1	bisection method . . . . .	13
Exa 2.2	bisection method . . . . .	14
Exa 2.3	bisection method . . . . .	14
Exa 2.4	bisection method . . . . .	15
Exa 2.5	bisection method . . . . .	16
Exa 2.6	false position method . . . . .	17
Exa 2.7	false position method . . . . .	18
Exa 2.8	false position method . . . . .	18
Exa 2.9	false position method . . . . .	19
Exa 2.10	iteration method . . . . .	20
Exa 2.11	iteration method . . . . .	21
Exa 2.12	iteration method . . . . .	21
Exa 2.13	iteration method . . . . .	22
Exa 2.14	aitkens process . . . . .	22

Exa 2.15	newton raphson method . . . . .	23
Exa 2.16	newton raphson method . . . . .	24
Exa 2.17	newton raphson method . . . . .	25
Exa 2.18	newton raphson method . . . . .	26
Exa 2.19	newton raphson method . . . . .	26
Exa 2.20	newton raphson method . . . . .	27
Exa 2.21	ramanujans method . . . . .	28
Exa 2.22	ramanujans method . . . . .	29
Exa 2.23	ramanujans method . . . . .	29
Exa 2.24	ramanujans method . . . . .	30
Exa 2.25	secant method . . . . .	31
Exa 2.26	secant method . . . . .	32
Exa 2.27	mullers method . . . . .	33
Exa 2.28	graeffes method . . . . .	34
Exa 2.29	quadratic factor by lins bairsttow method . . . . .	35
Exa 2.31	method of iteration . . . . .	36
Exa 2.32	newton raphson method . . . . .	36
Exa 2.33	newton raphson method . . . . .	37
Exa 2.34	newton raphson method . . . . .	38
Exa 3.4	interpolation . . . . .	40
Exa 3.6	interpolation . . . . .	41
Exa 3.7	interpolation . . . . .	42
Exa 3.8	interpolation . . . . .	43
Exa 3.9	Gauss forward formula . . . . .	45
Exa 3.10	practical interpolation . . . . .	47
Exa 3.11	practical interpolation . . . . .	48
Exa 3.12	practical interpolation . . . . .	49
Exa 3.13	lagranges interpolation formula . . . . .	50
Exa 3.14	lagranges interpolation formula . . . . .	51
Exa 3.15	lagranges interpolation formula . . . . .	52
Exa 3.16	lagranges interpolation formula . . . . .	52
Exa 3.17	lagranges interpolation . . . . .	53
Exa 3.18	error in lagranges interpolation formula . . . . .	54
Exa 3.19	error in lagranges interpolation formula . . . . .	55
Exa 3.21	hermites interpolation formula . . . . .	55
Exa 3.22	newtons general interpolation formula . . . . .	56
Exa 3.23	newtons divided formula . . . . .	57
Exa 3.24	interpolation by iteration . . . . .	58

Exa 3.25	inverse intrpolation . . . . .	58
Exa 3.26	double interpolation . . . . .	59
Exa 4.1	least square curve fitting procedure . . . . .	61
Exa 4.2	least square curve fitting procedure . . . . .	62
Exa 4.3	least square curve fitting procedure . . . . .	63
Exa 4.4	linearization of non linear law . . . . .	64
Exa 4.5	linearization of non linear law . . . . .	65
Exa 4.6	curve fitting by polynomial . . . . .	66
Exa 4.7	curve fitting by polynomial . . . . .	67
Exa 4.8	curve fitting by sum of exponentials . . . . .	69
Exa 4.9	linear weighted least approx . . . . .	72
Exa 4.10	linear weighted least approx . . . . .	73
Exa 4.11	least square for quadratic equations . . . . .	75
Exa 4.20	cooley Tukey method . . . . .	75
Exa 5.1	linear splines . . . . .	77
Exa 5.2	quadratic splines . . . . .	77
Exa 5.3	cubic splines . . . . .	78
Exa 5.4	cubic splines . . . . .	79
Exa 5.5	cubic splines . . . . .	80
Exa 5.7	surface fitting by cubic spline . . . . .	80
Exa 5.8	cubic B splines . . . . .	81
Exa 5.9	cubic B spline . . . . .	81
Exa 6.1	numerical diffrentiation by newtons difference formula . . . . .	83
Exa 6.2	numerical diffrentiation by newtons difference formula . . . . .	84
Exa 6.3	numerical diffrentiation by newtons difference formula . . . . .	86
Exa 6.4	estimation of errors . . . . .	87
Exa 6.5	cubic spline method . . . . .	89
Exa 6.6	derivative by cubic spline method . . . . .	90
Exa 6.7	maximun and minimun of functions . . . . .	90
Exa 6.8	trapezoidal method for integration . . . . .	91
Exa 6.9	simpson 1by3 method for integration . . . . .	91
Exa 6.10	integration by trapezoidal and simpsons method . . . . .	92
Exa 6.11	rommbergs method . . . . .	95
Exa 6.12	Trapezoidal and Simpsons rule . . . . .	97
Exa 6.13	area using cubic spline method . . . . .	98
Exa 6.15	eulers maclaurin formula . . . . .	99
Exa 6.17	error estimate in evaluation of the integral . . . . .	99
Exa 6.18	error estimate in evaluation of the integral . . . . .	100

Exa 6.19	gauss formula . . . . .	101
Exa 6.20	double integration . . . . .	101
Exa 7.1	inverse of matrix . . . . .	103
Exa 7.2	Factorize by triangulation method . . . . .	103
Exa 7.3	Vector Norms . . . . .	104
Exa 7.6	Gauss Jordan . . . . .	105
Exa 7.7	modern gauss jordan method . . . . .	106
Exa 7.8	LU decomposition method . . . . .	107
Exa 7.9	ill conditioned linear systems . . . . .	109
Exa 7.10	ill conditioned linear systems . . . . .	110
Exa 7.11	ill conditioned linear systems . . . . .	110
Exa 7.12	ill conditioned system . . . . .	111
Exa 7.14	solution of equations by iteration method . . . . .	111
Exa 7.15	eigenvalues and eigenvectors . . . . .	112
Exa 7.16	largest eigenvalue and eigenvectors . . . . .	113
Exa 7.17	housrholders method . . . . .	114
Exa 7.18	single value decommposition . . . . .	114
Exa 8.1	taylors method . . . . .	117
Exa 8.2	taylors method . . . . .	118
Exa 8.3	picards method . . . . .	119
Exa 8.4	picards method . . . . .	119
Exa 8.5	eulers method . . . . .	120
Exa 8.6	error estimates in eulers . . . . .	120
Exa 8.7	modified eulers method . . . . .	121
Exa 8.8	runge kutta formula . . . . .	122
Exa 8.9	runge kutta formula . . . . .	123
Exa 8.10	initial value problems . . . . .	124
Exa 8.11	adams moulton method . . . . .	124
Exa 8.12	milnes method . . . . .	125
Exa 8.13	milnes method . . . . .	127
Exa 8.14	initial value problems . . . . .	127
Exa 8.15	boundary value problem using finite difference method	128
Exa 8.16	boundary value problem using finite difference method	129
Exa 8.17	cubic spline method . . . . .	129
Exa 8.18	cubic spline method . . . . .	130
Exa 8.19	boundary value problem by cubic spline method . . .	130
Exa 9.1	standard five point formula . . . . .	132



Exa 9.2	solution of laplace equation by jacobi method gauss seidel method and SOR method . . . . .	132
Exa 9.4	poisson equation . . . . .	134
Exa 9.6	bender schmidt formula . . . . .	134
Exa 9.7	bender schimdts formula and crank nicolson formula .	135
Exa 9.8	heat equation using crank nicolson method . . . . .	136

# Chapter 1

## Errors in numerical calculation

Scilab code Exa 1.1 rounding off

```
1 //example 1.1
2 //rounding off
3 //page 7
4 clc;clear;close;
5 a1=1.6583;
6 a2=30.0567;
7 a3=0.859378;
8 a4=3.14159;
9 printf('\nthe numbers after rounding to 4
   significant figures are given below\n')
10 printf('    %f    %.4g\n',a1,a1);
11 printf('    %f    %.4g\n',a2,a2);
12 printf('    %f    %.4g\n',a3,a3);
13 printf('    %f    %.4g\n',a4,a4);
```

---

Scilab code Exa 1.2 percentage accuracy

```
1 //example 1.2
```

```

2 //percentage accuracy
3 //page 9
4 clc;clear;close;
5 x=0.51;// the number given
6 n=2;//correcting upto 2 decimal places
7 dx=((10^-n)/2)
8 p_a=(dx/x)*100;//percentage accuracy
9 printf('the percentage accuracy of %f after
    correcting to two decimal places is %f',x,p_a);

```

---

### Scilab code Exa 1.3 absolute and relative errors

```

1 //example 1.3
2 //absolute and relative errors
3 //page 9
4 clc;clear;close;
5 X=22/7;//approximate value of pi
6 T_X=3.1415926;// true value of pi
7 A_E=T_X-X;//absolute error
8 R_E=A_E/T_X;//relative error
9 printf('Absolute Error = %0.7f \n Relative Error =
    %0.7f ',A_E,R_E);

```

---

### Scilab code Exa 1.4 best approximation

```

1 //example 1.4
2 //best approximation
3 //page 10
4 clc;clear;close;
5 A_X=1/3;//the actual number
6 X1=0.30;
7 X2=0.33;
8 X3=0.34;

```

```

 9  A_E1=abs(A_X-X1);
10  A_E2=abs(A_X-X2);
11  A_E3=abs(A_X-X3);
12  if(A_E1<A_E2)
13  if(A_E1<A_E3)
14      B_A=X1;
15  end
16  end
17  if(A_E2<A_E1)
18  if(A_E2<A_E3)
19      B_A=X2;
20  end
21  end
22  if(A_E3<A_E2)
23  if(A_E3<A_E1)
24      B_A=X3;
25  end
26  end
27  printf('the best approximation of 1/3 is %.2g ',B_A)
    ;

```

---

#### Scilab code Exa 1.5 relative error

```

1  //relative error
2  //example 1.5
3  //page 10
4  clc;clear;close;
5  n=8.6;// the corrected number
6  N=1;//the no is rounded to one decimal places
7  E_A=(10^-N)/2;
8  E_R=E_A/n;
9  printf('the relative error of the number is:%0.4f',
    E_R);

```

---

### Scilab code Exa 1.6 absolute error and relative error

```
1 //example 1.6
2 //absolute error and relative error
3 //page 10
4 clc;clear;close;
5 s=sqrt(3)+sqrt(5)+sqrt(7);//the sum square root of
   3,5,7
6 n=4;
7 Ea=3*((10^-n)/2);//absolute error
8 R_E=Ea/s;
9 printf('the sum of square roots is %0.4g \n',s );
10 printf('the absolute error is %f \n',Ea);
11 printf('the relative error is %f ',R_E);
```

---

### Scilab code Exa 1.7 absolute error

```
1 //absolute error
2 //example 1.7
3 //page 10
4 clc;clear;close;
5 n=[0.1532 15.45 0.0000354 305.1 8.12 143.3 0.0212
   0.643 0.173];//original numbbbers
6 //rounding all numbers to 2 decimal places
7 n=[305.1 143.3 0.15 15.45 0.00 8.12 0.02 0.64];
8 sum=0;
9 l=length(n);
10 for i=1:l
11     sum=sum+n(i);
12 end
13 E_A=2*(10^-1)/2+7*(10^-2)/2;
14 printf('the absolute error is:%0.2f',E_A);
```

---

**Scilab code Exa 1.8** difference in 3 significant figures

```
1 //difference in 3 significant figures
2 //example 1.8
3 //page 11
4 clc;clear;close;
5 X1=sqrt(6.37);
6 X2=sqrt(6.36);
7 d=X1-X2;//difference between two numbers
8 printf('the differencecorrected to 3 significant
        figures is %0.3g',d);
```

---

**Scilab code Exa 1.10** relative error

```
1 //relative error
2 //example 1.10
3 //page 12
4 clc;clear;close;
5 a=6.54;b=48.64;c=13.5
6 da=0.01;db=0.02;dc=0.03;
7 s=(a^2*sqrt(b))/c^3;
8 disp(s,'s=');
9 r_err=2*(da/a)+(db/b)/2+3*(dc/c);
10 printf(' the relative error is :%f',r_err);
```

---

**Scilab code Exa 1.11** relative error

```
1 //relative error
2 //example 1.11
```

```

3 //page 13
4 clc;clear;close;
5 x=1;y=1;z=1;
6 u=(5*x*y^3)/z^3;
7 dx=0.001;dy=0.001;dz=0.001;
8 u_max=((5*y^2)/z^3)*dx+((10*x*y)/z^3)*dy+((15*x*y^2)
    /z^4)*dz;
9 r_err=u_max/u;
10 printf(' the relative error is :%f',r_err);

```

---

#### Scilab code Exa 1.12 taylor series

```

1 //taylor series
2 //example 1.12
3 //page 12
4 clc;clear;close;
5 deff('y=f(x)', 'y=x^3+5*x-10');
6 deff('y=f1(x)', 'y=3*x^2-6*x+5')//first derivative
7 deff('y=f2(x)', 'y=6*x-6')//second derivative
8 deff('y=f3(x)', 'y=6')//third derivative
9 D=[f(0) f1(0) f2(0) f3(0)]
10 S1=0;
11 h=1;
12 for i=1:4
13     S1=S1+h^(i-1)*D(i)/factorial(i-1);
14 end
15 printf('the third order taylor's series approximation
    of f(1) is :%d',S1);

```

---

#### Scilab code Exa 1.13 taylor series

```

1 //taylor series
2 //example 1.13

```

```

3 //page 16
4 clc;clear;close;
5 deff('y=f(x)', 'y=sin(x)');
6 deff('y=f1(x)', 'y=cos(x)');
7 deff('y=f2(x)', 'y=-sin(x)');
8 deff('y=f3(x)', 'y=-cos(x)');
9 deff('y=f4(x)', 'y=sin(x)');
10 deff('y=f5(x)', 'y=cos(x)');
11 deff('y=f6(x)', 'y=-sin(x)');
12 deff('y=f7(x)', 'y=-cos(x)');
13 D=[f(%pi/6) f1(%pi/6) f2(%pi/6) f3(%pi/6) f4(%pi/6)
      f5(%pi/6) f6(%pi/6) f7(%pi/6)];
14 S1=0;
15 h=%pi/6;
16 printf('order of approximation   computed value of
        sin(pi/3)   absolute error\n\n');
17 for j=1:8
18 for i=1:j
19     S1=S1+h^(i-1)*D(i)/factorial(i-1);
20 end
21 printf('%d                                %0.9 f
        %0.9 f\n', j, S1, abs(sin(%pi
        /3)-S1));
22 S1=0;
23 end

```

---

#### Scilab code Exa 1.14 maclaurins expansion

```

1 //maclaurins expansion
2 //example 1.14
3 //page 18
4 clc;clear;close;
5 x=1;
6 n=8;//correct to 8 decimal places
7 for i=1:50

```



```

8     if x/factorial(i)<(10^-8/2) then
9         c=i;
10        break;
11
12    end
13 end
14 printf('no of terms needed to correct to 8 decimal
        places is :%d',c)

```

---

### Scilab code Exa 1.15 series approximation

```

1 //series apprixamation
2 //example 1.15
3 //page 18
4 clc;clear;close;
5 x=1/11;
6 S1=0;
7 for i=1:2:5
8     S1=S1+(x^i/(i))
9     end
10 printf('value of log(1.2) is : %0.8f\n\n',2*S1)
11 c=0;
12 for i=1:50
13     if (1/11)^i/i<(2*10^-7) then
14         c=i;
15         break;
16     end
17 end
18 printf('min no of terms needed to get value wuth
        same accuracy is :%d',c)

```

---

## Chapter 2

# Solution of Algebraic and Transcendental Equation

Scilab code Exa 2.1 bisection method

```
1 //example 2.1
2 //bisection method
3 //page 24
4 clc;clear;close;
5 deff('y=f(x)', 'y=x^3-x-1');
6 x1=1,x2=2;//f(1) is negative and f(2) is positive
7 d=0.0001;//for accuracy of root
8 c=1;
9 printf('Successive approximations \t x1\t \tx2\t
\tm\t \tf(m)\n');
10 while abs(x1-x2)>d
11     m=(x1+x2)/2;
12     printf(' \t%f\t%f\t%f\t%f\n
\t,x1,x2,m,f(m));
13     if f(m)*f(x1)>0
14         x1=m;
15     else
16         x2=m;
17 end
```

```

18 c=c+1; // to count number of iterations
19 end
20 printf('the solution of equation after %i iteration
    is %g',c,m)

```

---

### Scilab code Exa 2.2 bisection method

```

1 //example 2.2
2 //bisection method
3 //page 25
4 clc;clear;close;
5 deff('y=f(x)', 'y=x^3-2*x-5');
6 x1=2,x2=3; //f(2) is negative and f(3) is positive
7 d=0.0001; //for accuracy of root
8 c=1;
9 printf('Successive approximations \t x1\t \tx2\t
    \tm\t \tf(m)\n');
10 while abs(x1-x2)>d
11     m=(x1+x2)/2;
12     printf(' \t%f\t%f\t%f\t%f\n
    ',x1,x2,m,f(m));
13     if f(m)*f(x1)>0
14         x1=m;
15     else
16         x2=m;
17     end
18 c=c+1; // to count number of iterations
19 end
20 printf('the solution of equation after %i iteration
    is %0.4g',c,m)

```

---

### Scilab code Exa 2.3 bisection method

```

1 //example 2.3
2 //bisection method
3 //page 26
4 clc;clear;close;
5 deff('y=f(x)', 'y=x^3+x^2+x+7');
6 x1=-3,x2=-2;//f(-3) is negative and f(-2) is
   positive
7 d=0.0001;//for accuracy of root
8 c=1;
9 printf('Successive approximations \t x1\t \tx2\t
   \tm\t \tf(m)\n');
10 while abs(x1-x2)>d
11     m=(x1+x2)/2;
12     printf('                \t%f\t%f\t%f\t%f\n
   ',x1,x2,m,f(m));
13     if f(m)*f(x1)>0
14         x1=m;
15     else
16         x2=m;
17 end
18 c=c+1;// to count number of iterations
19 end
20 printf('the solution of equation after %i iteration
   is %0.4g',c,m)

```

---

#### Scilab code Exa 2.4 bisection method

```

1 //example 2.4
2 //bisection method
3 //page 26
4 clc;clear;close;
5 deff('y=f(x)', 'y=x*exp(x)-1');
6 x1=0,x2=1;//f(0) is negative and f(1) is positive
7 d=0.0005;//maximun tolerance value
8 c=1;

```

```

9 printf('Successive approximations \t x1\t \tx2\t
\tm\t \ttol\t \tf(m)\n');
10 while abs((x2-x1)/x2)>d
11     m=(x1+x2)/2; //tolerance value for each iteration
12     tol=((x2-x1)/x2)*100;
13 printf(' \t%f\t%f\t%f\t%f\t
\t%f\n',x1,x2,m,tol,f(m));
14     if f(m)*f(x1)>0
15         x1=m;
16     else
17         x2=m;
18 end
19 c=c+1; // to count number of iterations
20 end
21 printf('the solution of equation after %i iteration
is %0.4g',c,m)

```

---

### Scilab code Exa 2.5 bisection method

```

1 //example 2.5
2 //bisection method
3 //page 27
4 clc;clear;close;
5 deff('y=f(x)', 'y=4*exp(-x)*sin(x)-1');
6 x1=0,x2=0.5; //f(0) is negative and f(1) is positive
7 d=0.0001; //for accuracy of root
8 c=1;
9 printf('Successive approximations \t x1\t \tx2\t
\tm\t \t \tf(m)\n');
10 while abs(x2-x1)>d
11     m=(x1+x2)/2;
12 printf(' \t%f\t%f\t%f\t%f\n
',x1,x2,m,f(m));
13     if f(m)*f(x1)>0
14         x1=m;

```

```

15     else
16         x2=m;
17     end
18     c=c+1; // to count number of iterations
19     end
20     printf('the solution of equation after %i iteration
           is %0.3g',c,m)

```

---

### Scilab code Exa 2.6 false position method

```

1 //example 2.6
2 //false position method
3 //page 28
4 clc;clear;close
5 deff('y=f(x)', 'y=x^3-2*x-5');
6 a=2,b=3; //f(2) is negative and f(3) is positive
7 d=0.00001;
8 printf('successive iterations      \ta\t      b\t      f(a
       )\t      f(b)\t\t      x1\n');
9 for i=1:25
10     x1=b*f(a)/(f(a)-f(b))+a*f(b)/(f(b)-f(a));
11     if(f(a)*f(x1))>0
12         b=x1;
13     else
14         a=x1;
15     end
16     if abs(f(x1))<d
17         break
18     end
19     printf('                \t%f      %f      %f
           %f      %f\n',a,b,f(a),f(b),x1);
20 end
21 printf('the root of the equation is %f',x1);

```

---

### Scilab code Exa 2.7 false position method

```
1 //example 2.7
2 //false position method
3 //page 29
4 clc;clear;close
5 deff('y=f(x)', 'y=x^2.2-69');
6 a=5,b=6;//f(5) is negative and f(6) is positive
7 d=0.00001;
8 printf('successive iterations      \ta\t      b\t      f(a
      )\t      f(b)\t\t      x1\n');
9 for i=1:25
10     x1=b*f(a)/(f(a)-f(b))+a*f(b)/(f(b)-f(a));
11     if(f(a)*f(x1))>0
12         b=x1;
13     else
14         a=x1;
15     end
16     if abs(f(x1))<d
17         break
18     end
19     printf('                \t%f   %f   %f
      %f   %f\n',a,b,f(a),f(b),x1);
20 end
21 printf('the root of the equation is   %f',x1);
```

---

### Scilab code Exa 2.8 false position method

```
1 //example 2.8
2 //false position method
3 //page 29
4 clc;clear;close
```

```

5 deff('y=f(x)', 'y=2*x-log10(x)-7');
6 a=3,b=4; //f(3) is negative and f(4) is positive
7 d=0.00001;
8 printf('successive iterations \ta\t b\t f(a
)\t f(b)\t\ x1\n');
9 for i=1:25
10 x1=b*f(a)/(f(a)-f(b))+a*f(b)/(f(b)-f(a));
11 if(f(a)*f(x1))>0
12 b=x1;
13 else
14 a=x1;
15 end
16 if abs(f(x1))<d
17 break
18 end
19 printf(' \t%f %f %f
%f %f\n', a, b, f(a), f(b), x1);
20 end
21 printf('the root of the equation is %0.4g', x1);

```

---

### Scilab code Exa 2.9 false position method

```

1 //example 2.9
2 //false position method
3 //page 30
4 clc;clear;close
5 deff('y=f(x)', 'y=4*exp(-x)*sin(x)-1');
6 a=0,b=0.5; //f(0) is negative and f(0.5) is positive
7 d=0.00001;
8 printf('successive iterations \ta\t b\t f(a
)\t f(b)\t\ x1\n');
9 for i=1:25
10 x1=b*f(a)/(f(a)-f(b))+a*f(b)/(f(b)-f(a));
11 if(f(a)*f(x1))>0
12 b=x1;

```



```

13     else
14         a=x1;
15     end
16     if abs(f(x1))<d
17         break
18     end
19     printf('          \t%f   %f   %f
          %f   %f\n',a,b,f(a),f(b),x1);
20 end
21 printf('the root of the equation is   %f',x1);

```

---

#### Scilab code Exa 2.10 iteration method

```

1 //example 2.10
2 //iteration method
3 //page 33
4 clc;clear,close;
5 deff('x=f(x)', 'x=1/(sqrt(x+1))');
6 x1=0.75,x2=0;
7 n=1;
8 d=0.0001;// accuracy upto 10^-4
9 c=0;// to count no of iterations
10 printf('successive iterations \t\x1\tf(x1)\n')
11 while abs(x1-x2)>d
12 printf('          \t%f      %f\n',x1,f(x1)
        )
13 x2=x1;
14 x1=f(x1);
15 c=c+1;
16 end
17 printf(' the root of the equation after %i iteration
        is %0.4g',c,x1)

```

---

### Scilab code Exa 2.11 iteration method

```
1 //example 2.11
2 //iteration method
3 //page34
4 clc;clear,close;
5 deff('x=f(x)', 'x=(cos(x)+3)/2');
6 x1=1.5; // as roots lies between 3/2 and pi/2
7 x2=0;
8 d=0.0001; // accuracy upto 10^-4
9 c=0; // to count no of iterations
10 printf('successive iterations \t\x1\tf(x1)\n')
11 while abs(x2-x1)>d
12 printf('          \t%f      %f\n',x1,f(x1)
13 )
14 x2=x1;
15 x1=f(x1);
16 c=c+1;
17 end
18 printf(' the root of the equation after %i iteration
19 is %0.4g',c,x1)
```

---

### Scilab code Exa 2.12 iteration method

```
1 //example 2.12
2 //iteration method
3 //page 35
4 clc;clear,close;
5 deff('x=f(x)', 'x=exp(-x)');
6 x1=1.5; // as roots lies between 0 and 1
7 x2=0;
8 d=0.0001; // accuracy upto 10^-4
9 c=0; // to count no of iterations
10 printf('successive iterations \t\x1\t f(x1)\n')
11 while abs(x2-x1)>d
```

```

12 printf('                \t%f      %f\n',x1,f(x1)
    )
13 x2=x1;
14 x1=f(x1);
15 c=c+1;
16 end
17 printf(' the root of the eqaution after %i iteration
    is %0.4g',c,x1)

```

---

### Scilab code Exa 2.13 iteration method

```

1 //example 2.12
2 //iteration method
3 //page 35
4 clc;clear,close;
5 deff('x=f(x)', 'x=1+(sin(x)/10)');
6 x1=1;// as roots lies between 1 and pi evident from
    graph
7 x2=0;
8 d=0.0001;// accuracy upto 10-4
9 c=0;// to count no of iterations
10 printf('successive iterations \t\x1\tf(x1)\n')
11 while abs(x2-x1)>d
12 printf('                \t%f      %f\n',x1,f(x1))
13 x2=x1;
14 x1=f(x1);
15 c=c+1;
16 end
17 printf(' the root of the eqaution after %i iteration
    is %0.4g',c,x1)

```

---

### Scilab code Exa 2.14 aitkens process

```

1 //example 2.14
2 //aitken's process
3 //page 36
4 clc,clear,close
5 deff('x=f(x)', 'x=(3+cos(x))/2');
6 x0=1.5;
7 y=0;
8 e=0.0001;
9 c=0;
10 printf('successive iterations      \tx0\t      x1\t
        x2\t      x3\t      y\n')
11 for i=1:10
12     x1=f(x0),x2=f(x1),x3=f(x2);
13     y=x3-((x3-x2)^2)/(x3-2*x2+x1);
14     d=y-x0;
15     x0=y;
16     if abs(f(x0))<e then
17         break;
18     end
19     c=c+1;
20 printf('      \t%f      %f      %f
        %f      %f\n',x0,x1,x2,x3,y)
21 end
22 printf('the root of the equation after %i iteration
        is %f',c,y);

```

---

**Scilab code Exa 2.15** newton raphson method

```

1 //example 2.15
2 //newton-raphson method
3 //page 39
4 clc;clear;close
5 deff('y=f(x)', 'y=x^3-2*x-5');
6 deff('y1=f1(x)', 'y1=3*x^2-2');// first derivative of
        the function

```

```

7 x0=2; // initial value
8 d=0.0001;
9 c=0;n=1;
10 printf('successive iterations \tx0\t          f(x0)\t
          f1(x0)\n');
11 while n==1
12     x2=x0;
13     x1=x0-(f(x0)/f1(x0));
14     x0=x1;
15 printf('          \t%f\t%f\t%f\n',x2,f(
          x1),f1(x1))
16 c=c+1;
17 if abs(f(x0))<d then
18 break;
19 end
20 end
21 printf('the root of %i iteration is:%f',c,x0);

```

---

### Scilab code Exa 2.16 newton raphson method

```

1 //example 2.16
2 //newton-raphson method
3 //page 40
4 clc;clear;close
5 deff('y=f(x)', 'y=x*sin(x)+cos(x)');
6 deff('y1=f1(x)', 'y1=x*cos(x)');//first derivation of
  the function
7 x0=%pi; // initial value
8 d=0.0001;
9 c=0;n=1;
10 printf('successive iterations \tx0\t          f(x0)\t
          f1(x0)\n');
11 while n==1
12     x2=x0
13     x1=x0-(f(x0)/f1(x0));

```

```

14     x0=x1;
15     printf('                \t%f\t%f\t%f\n',x2 , f(
        x1),f1(x1))
16     c=c+1;
17     if abs(f(x0))<d then
18     break;
19     end
20     end
21     printf('the root of %i iteration is:%f',c,x0);

```

---

### Scilab code Exa 2.17 newton raphson method

```

1 //example 2.17
2 //newton-raphson method
3 //page 40
4 clc;clear;close
5 deff('y=f(x)', 'y=x*exp(x)-1');
6 deff('y1=f1(x)', 'y1=exp(x)+x*exp(x)'); // first
    derivative of the function
7 x0=0; // initial value
8 d=0.0001;
9 c=0;n=1
10 printf('successive iterations \tx0\t                f(x0)\t
        f1(x0)\n');
11 while n==1
12     x2=x0;
13     x1=x0-(f(x0)/f1(x0));
14     x0=x1;
15     printf('                \t%f\t%f\t%f\n',x2 , f(
        x1),f1(x1))
16     c=c+1;
17     if abs(f(x0))<d then
18     break;
19     end
20     end

```

```
21 printf('the root of %i iteration is:%f',c,x0);
```

---

**Scilab code Exa 2.18** newton raphson method

```
1 //example 2.18
2 //newton-raphson method
3 //page 41
4 clc;clear;close
5 deff('y=f(x)', 'y=sin(x)-x/2');
6 deff('y1=f1(x)', 'y1=cos(x)-1/2');
7 x0=%pi/2; // initial value
8 d=0.0001;
9 c=0;n=1;
10 printf('successive iterations \tx0\t          f(x0)\t
          f1(x0)\n');
11 while n==1
12     x2=x0;
13     x1=x0-(f(x0)/f1(x0));
14     x0=x1;
15
16 printf('          \t%f\t%f\t%f\n',x2,f(
          x1),f1(x1))
17 c=c+1;
18 if abs(f(x0))<d then
19     break;
20 end
21 end
22 printf('the root of %i iteration is:%0.4g',c,x0);
```

---

**Scilab code Exa 2.19** newton raphson method

```
1 //example 2.19
2 //newton-raphson method
```

```

3 //page 41
4 clc;clear;close
5 deff('y=f(x)', 'y=4*exp(-x)*sin(x)-1');
6 deff('y1=f1(x)', 'y1=cos(x)*4*exp(-x)-4*exp(-x)*sin(x)');
7 x0=0.2; // initial value
8 d=0.0001;
9 c=0;n=1;
10 printf('successive iterations \tx0\t          f(x0)\t
          f1(x0)\n');
11 while n==1
12     x2=x0;
13     x1=x0-(f(x0)/f1(x0));
14     x0=x1;
15     printf('          \t%f\t%f\t%f\n',x2,f(
          x1),f1(x1))
16     c=c+1;
17     if abs(f(x0))<d then
18         break;
19     end
20     end
21     printf('the root of %i iteration is:%0.3g',c,x0);

```

---

### Scilab code Exa 2.20 newton raphson method

```

1 //example 2.20
2 //generalized newton-raphson method
3 //page 42
4 clc;clear;close;
5 deff('y=f(x)', 'y=x^3-x^2-x+1');
6 deff('y1=f1(x)', 'y1=3*x^2-2*x-1');
7 deff('y2=f2(x)', 'y2=6*x-2');
8 x0=0.8; // initial value to find double root
9 n=1;
10 printf('successive iterations \tx0\t          x1\t

```



```

                x2\n')
11 while n==1
12 x1=x0-(f(x0)/f1(x0));
13 x2=x0-(f1(x0)/f2(x0));
14 if abs(x1-x2)<0.000000001 then
15     x0=(x1+x2)/2;
16     break;
17 else
18     x0=(x1+x2)/2;
19 end
20 printf('                %f\t%f\t%f\n',x0,
        x1,x2);
21 end
22 printf('\n \nthe double root is at: %f',x0 );

```

---

### Scilab code Exa 2.21 ramanujans method

```

1 //ramanujan's method
2 //example 2.21
3 //page 45
4 clc;clear;close;
5 deff('y=f(x)', '1-((13/12)*x-(3/8)*x^2+(1/24)*x^3)');
6 a1=13/12, a2=-3/8, a3=1/24;
7 b1=1;
8 b2=a1;
9 b3=a1*b2+a2*b1;
10 b4=a1*b3+a2*b2+a3*b1;
11 b5=a1*b4+a2*b3+a3*b2;
12 b6=a1*b5+a2*b4+a3*b3;
13 b7=a1*b6+a2*b5+a3*b4;
14 b8=a1*b7+a2*b6+a3*b5;
15 b9=a1*b8+a2*b7+a3*b6;
16 printf('\n\n%f', b1/b2);
17 printf('\n%f', b2/b3);
18 printf('\n%f', b3/b4);

```

```

19 printf( '\n%f', b4/b5);
20 printf( '\n%f', b5/b6);
21 printf( '\n%f', b6/b7);
22 printf( '\n%f', b7/b8);
23 printf( '\n%f', b8/b9);
24 printf( '\n it appears as if the roots are converging
      at 2')

```

---

### Scilab code Exa 2.22 ramanujans method

```

1 //ramanujan's method
2 //example 2.22
3 //page 46
4 clc;clear;close;
5 deff('y=f(x)', 'x+x^2+x^3/2+x^4/6+x^5/24');
6 a1=1, a2=1, a3=1/2, a4=1/6, a5=1/24;
7 b1=1;
8 b2=a2;
9 b3=a1*b2+a2*b1;
10 b4=a1*b3+a2*b2+a3*b1;
11 b5=a1*b4+a2*b3+a3*b2;
12 b6=a1*b5+a2*b4+a3*b3;
13 printf( '\n%f', b1/b2);
14 printf( '\n%f', b2/b3);
15 printf( '\n%f', b3/b4);
16 printf( '\n%f', b4/b5);
17 printf( '\n%f', b5/b6);
18 printf( '\n it appears as if the roots are converging
      at around %f', b5/b6);

```

---

### Scilab code Exa 2.23 ramanujans method

```

1 //ramanujan's method

```

```

2 //example 2.23
3 //page 47
4 clc;clear;close;
5 deff('y=f(x)', '1-2*((3/2)*x+(1/4)*x^2-(1/48)*x^4+x
    ^6/1440-x^8/80640)');
6 a1=3/2, a2=1/4, a3=0, a4=1/48, a5=0, a6=1/1440, a7=0, a8
    =-1/80640;
7 b1=1;
8 b2=a1;
9 b3=a1*b2+a2*b1;
10 b4=a1*b3+a2*b2+a3*b1;
11 b5=a1*b4+a2*b3+a3*b2;
12 b6=a1*b5+a2*b4+a3*b3;
13 b7=a1*b6+a2*b5+a3*b4;
14 b8=a1*b7+a2*b6+a3*b5;
15 b9=a1*b8+a2*b7+a3*b6;
16 printf('\n%f', b1/b2);
17 printf('\n%f', b2/b3);
18 printf('\n%f', b3/b4);
19 printf('\n%f', b4/b5);
20 printf('\n%f', b5/b6);
21 printf('\n%f', b6/b7);
22 printf('\n%f', b7/b8);
23 printf('\n it appears as if the roots are converging
    at around %f', b7/b8)

```

---

#### Scilab code Exa 2.24 ramanujans method

```

1 //ramanujan's method
2 //example 2.23
3 //page 47
4 clc;clear;close;
5 deff('y=f(x)', '1-(x-x^2/factorial(2)^2+x^3/factorial
    (3)^2-x^4/factorial(4)^2)');
6 a1=1, a2=-1/(factorial(2)^2), a3=1/(factorial(3)^2), a4

```

```

    =-1/(factorial(4)^2), a5=-1/(factorial(5)^2), a6
    =1/(factorial(6)^2);
7 b1=1;
8 b2=a1;
9 b3=a1*b2+a2*b1;
10 b4=a1*b3+a2*b2+a3*b1;
11 b5=a1*b4+a2*b3+a3*b2;
12 printf( '\n\n%f', b1/b2);
13 printf( '\n%f', b2/b3);
14 printf( '\n%f', b3/b4);
15 printf( '\n%f', b4/b5);
16 printf( '\n it appears as if the roots are converging
    at around %f', b4/b5);

```

---

### Scilab code Exa 2.25 secant method

```

1 //example 2.25
2 //secant method
3 //page 49
4 clc;clear;close;
5 deff( 'y=f(x)', 'y=x^3-2*x-5' );
6 x1=2,x2=3// initial values
7 n=1;
8 c=0;
9 printf( 'successive iterations      \tx1          \tx2\
    t          x3\t          f(x3)\n' )
10 while n==1
11     x3=(x1*f(x2)-x2*f(x1))/(f(x2)-f(x1));
12     printf( '          \t%f\t%f\t%f\t%f\n
    ',x1,x2,x3,f(x3));
13     if f(x3)*f(x1)>0 then
14         x2=x3;
15     else
16         x1=x3;
17     end

```

```

18 if abs(f(x3)) < 0.000001 then
19     break;
20 end
21 c=c+1;
22 end
23 printf('the root of the equation after %i iteration
        is: %f', c, x3 )

```

---

### Scilab code Exa 2.26 secant method

```

1 //example 2.26
2 //secant method
3 //page 50
4 clc; clear; close;
5 deff('y=f(x)', 'y=x*exp(x)-1');
6 x1=0, x2=1 // initial values
7 n=1;
8 c=0;
9 printf('successive iterations      \tx1          \tx2\
        t          x3\t          f(x3)\n')
10 while n==1
11     x3=(x1*f(x2)-x2*f(x1))/(f(x2)-f(x1));
12 printf('                        \t%f\t%f\t%f\t%f\n
        ', x1, x2, x3, f(x3));
13 if f(x3)*f(x1) > 0 then
14     x2=x3;
15 else
16     x1=x3;
17 end
18 if abs(f(x3)) < 0.0001 then
19     break;
20 end
21 c=c+1;
22 end
23 printf('the root of the equation after %i iteration

```

is : %0.4g',c,x3 )

---

### Scilab code Exa 2.27 mullers method

```
1 // example 2.27
2 //muller's method
3 //page 52
4 clc;clear;close;
5 deff ('y=f(x)', 'y=x^3-x-1');
6 x0=0,x1=1,x2=2; // initial values
7 n=1;c=0;
8 printf(' successive iterations      \tx0\t      x1\t
          x2\t      f(x0)\t      f(x1)\t      f(x2)\n')
9 while n==1
10     c=c+1;
11     y0=f(x0),y1=f(x1),y2=f(x2);
12     h2=x2-x1,h1=x1-x0;
13     d2=f(x2)-f(x1),d1=f(x1)-f(x0);
14     printf('                \t%f\t      %f\t      %f
          \t %f\t      %f\t      %f\n',x0,x1,x2,f(x0),f(x1),f(x2))
          ;
15     A=(d2/h2-d1/h1)/(h1+h2);
16     B=d2/h2+A*h2;;
17     S=sqrt(B^2-4*A*f(x2));
18     x3=x2-(2*f(x2))/(B+S);
19     E=abs((x3-x2)/x2)*100;
20     if E<0.003 then
21         break;
22     else
23         if c==1 then
24             x2=x3;
25         end
26     if c==2 then
27         x1=x2;
28         x2=x3;
```

```

29 end
30 if c==3 then
31     x0=x1;
32     x1=x2;
33     x2=x3;
34 end
35 if c==3 then
36     c=0;
37 end
38 end
39 end
40 printf('the required root is : %0.4f',x3)

```

---

#### Scilab code Exa 2.28 graeffes method

```

1 //graeffe 's method
2 //example 2.28
3 //page 55
4 clc;clear;close;
5 deff('y=f(x)', 'y=x^3-6*x^2+11*x-6');
6 x=poly(0, 'x');
7 g=f(-x);
8 printf('the equation is:\n');
9 disp(g(x)*f(x));
10 A=[1 14 49 36]; //coefficients of the above equation
11 printf('%0.4g\n',sqrt(A(4)/A(3)));
12 printf('%0.4g\n',sqrt(A(3)/A(2)));
13 printf('%0.4g\n',sqrt(A(2)/A(1)));
14 printf('the equation is:\n');
15 disp(g*(-1*g));
16 B=[1 98 1393 1296];
17 printf('%0.4g\n', (B(4)/B(3))^(1/4));
18 printf('%0.4g\n', (B(3)/B(2))^(1/4));
19 printf('%0.4g\n', (B(2)/B(1))^(1/4));
20 printf('It is apparent from the outputs that the

```

roots converge at 1 2 3')

---

**Scilab code Exa 2.29** quadratic factor by lins bairsttow method

```
1 //quadratic factor by lin 's—bairsttow method
2 //example 2.29
3 //page 57
4 clc;clear;close;
5 deff('y=f(x)', 'y=x^3-x-1');
6 a=[-1 -1 0 1];
7 r1=1;s1=1;
8 b4=a(4);
9 deff('b3=f3(r)', 'b3=a(3)-r*a(4)');
10 deff('b2=f2(r,s)', 'b2=a(2)-r*a(3)+r^2*a(4)-s*a(4)');
11 deff('b1=f1(r,s)', 'b1=a(1)-s*a(3)+s*r*a(4)');
12 A=[1,1;2,-1];
13 C=[0;1];
14 X=A^-1*C;
15 dr=X(1,1);ds=X(2,1);
16 r2=r1+dr;s2=s1+ds;
17 //second pproximation
18 r1=r2;s1=s2;
19 b11=f1(r2,s2);
20 b22=f2(r2,s2);
21 h=0.001;
22 dr_b1=(f1(r1+h,s1)-f1(r1,s1))/h;
23 ds_b1=(f1(r1,s1+h)-f1(r1,s1))/h;
24 dr_b2=(f2(r1+h,s1)-f2(r1,s1))/h;
25 ds_b2=(f2(r1,s1+h)-f2(r1,s1))/h;
26 A=[dr_b1,ds_b1;dr_b2,ds_b2];
27 C=[-f1(r1,s1);-f2(r1,s2)];
28 X=A^-1*C;
29 r2=r1+X(1,1);
30 s2=s1+X(2,1);
31 printf(' roots correct to 3 decimal places are : %0
```



```
.3 f          %0.3 f ',r2,s2);
```

---

**Scilab code Exa 2.31** method of iteration

```
1 //method of iteration
2 //example 2.31
3 //page 62
4 clc;clear;close;
5 deff('x=f(x,y)', '(3*y*x^2+7)/10');
6 deff('y=g(x,y)', '(y^2+4)/5');
7 h=0.0001;
8 x0=0.5;y0=0.5;
9 f1_dx=(f(x0+h,y0)-f(x0,y0))/h;
10 f1_dy=(f(x0,y0+h)-f(x0,y0))/h;
11 g1_dx=(g(x0+h,y0)-g(x0,y0))/h;
12 g1_dy=(g(x0+h,y0)-g(x0,y0))/h;
13 if f1_dx+f1_dy<1 & g1_dx+g1_dy<1 then
14     printf('conditions for convergence is satisfied\n
15           \n')
16 end
17 printf(' X\t          Y\t\n\n');
18 for i=1:10
19     X=(3*y0*x0^2+7)/10;
20     Y=(y0^2+4)/5;
21     printf('%f\t          %f\t\n',X,Y);
22     x0=X;y0=Y;
23 end
24 printf('\n\n CONVERGENCE AT (1 1) IS OBVIOUS');
```

---

**Scilab code Exa 2.32** newton raphson method

```
1 //newton raphson method
2 //example 2.32
```

```

3 //page 65
4 clc;clear;close;
5 deff('y=f(x,y)', 'y=3*y*x^2-10*x+7');
6 deff('x=g(y)', 'x=y^2-5*y+4');
7 hh=0.0001;
8 x0=0.5,y0=0.5; //initial values
9 f0=f(x0,y0);
10 g0=g(y0);
11 df_dx=(f(x0+hh,y0)-f(x0,y0))/hh;
12 df_dy=(f(x0,y0+hh)-f(x0,y0))/hh;
13 dg_dx=(g(y0)-g(y0))/hh;
14 dg_dy=(g(y0+hh)-g(y0))/hh;
15 D1=determ([df_dx,df_dy;dg_dx,dg_dy]);
16 h=determ([-f0,df_dy;-g0,dg_dy])/D1;
17 k=determ([df_dx,-f0;dg_dx,-g0])/D1;
18 x1=x0+h;
19 y1=y0+k;
20 f0=f(x1,y1);
21 g0=g(y1);
22 df_dx=(f(x1+hh,y1)-f(x1,y1))/hh;
23 df_dy=(f(x1,y1+hh)-f(x1,y1))/hh;
24 dg_dx=(g(y1)-g(y1))/hh;
25 dg_dy=(g(y1+hh)-g(y1))/hh;
26 D2=determ([df_dx,df_dy;dg_dx,dg_dy]);
27 h=determ([-f0,df_dy;-g0,dg_dy])/D2;
28 k=determ([df_dx,-f0;dg_dx,-g0])/D2;
29 x2=x1+h;
30 y2=y1+k;
31 printf(' the roots of the equation are x2=%f and y2=
%f ',x2,y2);

```

---

### Scilab code Exa 2.33 newton raphson method

```

1 //newton raphson method
2 //example 2.33

```

```

3 //page 66
4 clc;clear;close;
5 deff('y=f(x,y)', 'y=x^2+y^2-1');
6 deff('x=g(x,y)', 'x=y-x^2');
7 hh=0.0001;
8 x0=0.7071, y0=0.7071; //initial values
9 f0=f(x0, y0);
10 g0=g(x0, y0);
11 df_dx=(f(x0+hh, y0)-f(x0, y0))/hh;
12 df_dy=(f(x0, y0+hh)-f(x0, y0))/hh;
13 dg_dx=(g(x0+hh, y0)-g(x0, y0))/hh;
14 dg_dy=(g(x0, y0+hh)-g(x0, y0))/hh;
15 D1=determ([df_dx, df_dy; dg_dx, dg_dy]);
16 h=determ([-f0, df_dy; -g0, dg_dy])/D1;
17 k=determ([df_dx, -f0; dg_dx, -g0])/D1;
18 x1=x0+h;
19 y1=y0+k;
20 f0=f(x1, y1);
21 g0=g(x1, y1);
22 df_dx=(f(x1+hh, y1)-f(x1, y1))/hh;
23 df_dy=(f(x1, y1+hh)-f(x1, y1))/hh;
24 dg_dx=(g(x1+hh, y1)-g(x1, y1))/hh;
25 dg_dy=(g(x1, y1+hh)-g(x1, y1))/hh;
26 D2=determ([df_dx, df_dy; dg_dx, dg_dy]);
27 h=determ([-f0, df_dy; -g0, dg_dy])/D2;
28 k=determ([df_dx, -f0; dg_dx, -g0])/D2;
29 x2=x1+h;
30 y2=y1+k;
31 printf(' the roots of the equation are x2=%f and y2=
%f ', x2, y2);

```

---

Scilab code Exa 2.34 newton raphson method

```

1 //newton raphson method
2 //example 2.33

```

```

3 //page 66
4 clc;clear;close;
5 deff('y=f(x,y)', 'y=sin(x)-y+0.9793');
6 deff('x=g(x,y)', 'x=cos(y)-x+0.6703');
7 hh=0.0001;
8 x0=0.5,y0=1.5; //initial values
9 f0=f(x0,y0);
10 g0=g(x0,y0);
11 df_dx=(f(x0+hh,y0)-f(x0,y0))/hh;
12 df_dy=(f(x0,y0+hh)-f(x0,y0))/hh;
13 dg_dx=(g(x0+hh,y0)-g(x0,y0))/hh;
14 dg_dy=(g(x0,y0+hh)-g(x0,y0))/hh;
15 D1=determ([df_dx,df_dy;dg_dx,dg_dy]);
16 h=determ([-f0,df_dy;-g0,dg_dy])/D1;
17 k=determ([df_dx,-f0;dg_dx,-g0])/D1;
18 x1=x0+h;
19 y1=y0+k;
20 f0=f(x1,y1);
21 g0=g(x1,y1);
22 df_dx=(f(x1+hh,y1)-f(x1,y1))/hh;
23 df_dy=(f(x1,y1+hh)-f(x1,y1))/hh;
24 dg_dx=(g(x1+hh,y1)-g(x1,y1))/hh;
25 dg_dy=(g(x1,y1+hh)-g(x1,y1))/hh;
26 D2=determ([df_dx,df_dy;dg_dx,dg_dy]);
27 h=determ([-f0,df_dy;-g0,dg_dy])/D2;
28 k=determ([df_dx,-f0;dg_dx,-g0])/D2;
29 x2=x1+h;
30 y2=y1+k;
31 printf(' the roots of the equation are x2=%0.4f and
        y2=%0.4f ',x2,y2);

```

---

# Chapter 3

## interpolation

Scilab code Exa 3.4 interpolation

```
1 //example 3.4
2 //interpolation
3 //page 86
4 clc;clear;close;
5 x=[1 3 5 7];
6 y=[24 120 336 720];
7 h=2//interval between values of x
8 c=1;
9 for i=1:3
10     d1(c)=y(i+1)-y(i);
11     c=c+1;
12 end
13 c=1;
14 for i=1:2
15     d2(c)=d1(i+1)-d1(i);
16     c=c+1
17 end
18 c=1;
19 for i=1:1
20     d3(c)=d2(i+1)-d2(i);
21     c=c+1;
```

```

22 end
23
24 d=[d1(1) d2(1) d3(1)];
25 x0=8; //value at 8;
26 pp=1;
27 y_x=y(1);
28 p=(x0-1)/2;
29 for i=1:3
30     pp=1;
31     for j=1:i
32         pp=pp*(p-(j-1))
33     end
34 y_x=y_x+(pp*d(i))/factorial(i);
35 end
36 printf('value of function at %f is :%f',x0,y_x);

```

---

### Scilab code Exa 3.6 interpolation

```

1 //example 3.6
2 //interpolation
3 //page 87
4 clc;clear;close;
5 x=[15 20 25 30 35 40];
6 y=[0.2588190 0.3420201 0.4226183 0.5 0.5735764
    0.6427876];
7 h=5//interval between values of x
8 c=1;
9 for i=1:5
10     d1(c)=y(i+1)-y(i);
11     c=c+1;
12 end
13 c=1;
14 for i=1:4
15     d2(c)=d1(i+1)-d1(i);
16     c=c+1

```

```

17 end
18 c=1;
19 for i=1:3
20     d3(c)=d2(i+1)-d2(i);
21     c=c+1;
22 end
23 c=1;
24 for i=1:2
25     d4(c)=d3(i+1)-d3(i);
26     c=c+1;
27 end
28 c=1;
29 for i=1:1
30     d5(c)=d4(i+1)-d4(i);
31     c=c+1;
32 end
33 c=1;
34 d=[d1(5) d2(4) d3(3) d4(2) d5(1)];
35 x0=38; //value at 38 degree
36 pp=1;
37 y_x=y(6);
38 p=(x0-x(6))/h;
39 for i=1:5
40     pp=1;
41     for j=1:i
42         pp=pp*(p+(j-1))
43     end
44 y_x=y_x+((pp*d(i))/factorial(i));
45 end
46 printf('value of function at %i is :%f',x0,y_x);

```

---

Scilab code Exa 3.7 interpolation

```

1 //example 3.7
2 //interpolation

```

```

3 //page 89
4 clc;clear;close;
5 x=[0 1 2 4];
6 y=[1 3 9 81];
7 //equation is  $y(5)-4*y(4)+6*y(2)-4*y(2)+y(1)$ 
8 y3=(y(4)+6*y(3)-4*y(2)+y(1))/4;
9 printf(' the value of missing term of table is :%d',
        y3);

```

---

### Scilab code Exa 3.8 interpolation

```

1 //example 3.8
2 //interpolation
3 //page 89
4 clc;clear;close;
5 x=[0.10 0.15 0.20 0.25 0.30];
6 y=[0.1003 0.1511 0.2027 0.2553 0.3093];
7 h=0.05//interval between values of x
8 c=1;
9 for i=1:4
10     d1(c)=y(i+1)-y(i);
11     c=c+1;
12 end
13 c=1;
14 for i=1:3
15     d2(c)=d1(i+1)-d1(i);
16     c=c+1
17 end
18 c=1;
19 for i=1:2
20     d3(c)=d2(i+1)-d2(i);
21     c=c+1;
22 end
23 c=1;
24 for i=1:1

```



```

25     d4(c)=d3(i+1)-d3(i);
26     c=c+1;
27 end
28
29 d=[d1(1) d2(1) d3(1) d4(1)];
30 x0=0.12; //value at 0.12;
31 pp=1;
32 y_x=y(1);
33 p=(x0-x(1))/h;
34 for i=1:4
35     pp=1;
36     for j=1:i
37         pp=pp*(p-(j-1))
38     end
39 y_x=y_x+(pp*d(i))/factorial(i);
40 end
41 printf('value of function at %f is :%0.4g\n \n',x0,
        y_x);
42 d=[d1(4) d2(3) d3(2) d4(1)];
43 x0=0.26; //value at 0.26;
44 pp=1;
45 y_x=y(5);
46 p=(x0-x(5))/h;
47 for i=1:4
48     pp=1;
49     for j=1:i
50         pp=pp*(p-(j-1))
51     end
52 y_x=y_x+(pp*d(i))/factorial(i);
53 end
54 printf('value of function at %f is :%0.4g\n \n',x0,
        y_x);
55 d=[d1(4) d2(3) d3(2) d4(1)];
56 x0=0.40; //value at 0.40;
57 pp=1;
58 y_x=y(5);
59 p=(x0-x(5))/h;
60 for i=1:4

```

```

61     pp=1;
62     for j=1:i
63         pp=pp*(p+(j-1))
64     end
65     y_x=y_x+(pp*d(i))/factorial(i);
66 end
67 printf('value of function at %f is :%0.4g\n \n',x0,
        y_x);
68 d=[d1(4) d2(3) d3(2) d4(1)];
69 x0=0.50; //value at 0.50;
70 pp=1;
71 y_x=y(5);
72 p=(x0-x(5))/h;
73 printf('value of function at %f is :%0.5g\n \n',x0,
        y_x);

```

---

### Scilab code Exa 3.9 Gauss forward formula

```

1 //example 3.9
2 //Gauss' forward formula
3 //page 3.9
4 clc;clear;close;
5 x=[1.0 1.05 1.10 1.15 1.20 1.25 1.30];
6 y=[2.7183 2.8577 3.0042 3.1582 3.3201 3.4903
    3.66693];
7 h=0.05//interval between values of x
8 c=1;
9 for i=1:6
10     d1(c)=y(i+1)-y(i);
11     c=c+1;
12 end
13 c=1;
14 for i=1:5
15     d2(c)=d1(i+1)-d1(i);
16     c=c+1

```

```

17 end
18 c=1;
19 for i=1:4
20     d3(c)=d2(i+1)-d2(i);
21     c=c+1;
22 end
23 c=1;
24 for i=1:3
25     d4(c)=d3(i+1)-d3(i);
26     c=c+1;
27 end
28 c=1;
29 for i=1:2
30     d5(c)=d4(i+1)-d4(i);
31     c=c+1;
32 end
33 c=1;
34 for i=1:1
35     d6(c)=d5(i+1)-d5(i);
36     c=c+1;
37 end
38 d=[d1(4) d2(3) d3(3) d4(2) d5(1) d6(1)];
39 x0=1.17; //value at 1.17;
40 pp=1;
41 y_x=y(4);
42 p=(x0-x(4))/h;
43 for i=1:6
44     pp=1;
45     for j=1:i
46         pp=pp*(p-(j-1))
47     end
48 y_x=y_x+(pp*d(i))/factorial(i);
49 end
50 printf('value of function at %f is :%0.4g\n \n',x0,
        y_x);

```

---

### Scilab code Exa 3.10 practical interpolation

```
1 //practical interpolation
2 //example 3.10
3 //page 97
4 clc;clear;close;
5 x=[0.61 0.62 0.63 0.64 0.65 0.66 0.67];
6 y=[1.840431 1.858928 1.877610 1.896481 1.915541
    1.934792 1.954237];
7 h=0.01//interval between values of x
8 c=1;
9 for i=1:6
10     d1(c)=y(i+1)-y(i);
11     c=c+1;
12 end
13 c=1;
14 for i=1:5
15     d2(c)=d1(i+1)-d1(i);
16     c=c+1
17 end
18 c=1;
19 for i=1:4
20     d3(c)=d2(i+1)-d2(i);
21     c=c+1;
22 end
23 c=1;
24 for i=1:3
25     d4(c)=d3(i+1)-d3(i);
26     c=c+1;
27 end
28 d=[d1(1) d2(1) d3(1) d4(1)];
29 x0=0.644;
30 p=(x0-x(4))/h;
31 y_x=y(4);
```

```

32 y_x=y_x+p*(d1(3)+d1(4))/2+p^2*(d2(2))/2; //stirling
    formula
33 printf(' the value at %f by stirling formula is : %f
    \n\n',x0,y_x);
34 y_x=y(4);
35 y_x=y_x+p*d1(4)+p*(p-1)*(d2(3)+d2(4))/2;
36 printf(' the value at %f by bessels formula is : %f\
    n\n',x0,y_x);
37 y_x=y(4);
38 q=1-p;
39 y_x=q*y(4)+q*(q^2-1)*d2(3)/2+p*y(5)+p*(q^2-1)*d2(4)
    /2;
40 printf(' the value at %f by everrets formula is : %f
    \n\n',x0,y_x);

```

---

### Scilab code Exa 3.11 practical interpolation

```

1 //practical interpolation
2 //example 3.11
3 //page 99
4 clc;clear;close;
5 x=[0.61 0.62 0.63 0.64 0.65 0.66 0.67];
6 y=[1.840431 1.858928 1.877610 1.896481 1.915541
    1.934792 1.954237];
7 h=0.01//interval between values of x
8 c=1;
9 for i=1:6
10     d1(c)=y(i+1)-y(i);
11     c=c+1;
12 end
13 c=1;
14 for i=1:5
15     d2(c)=d1(i+1)-d1(i);
16     c=c+1
17 end

```

```

18 c=1;
19 for i=1:4
20     d3(c)=d2(i+1)-d2(i);
21     c=c+1;
22 end
23 c=1;
24 for i=1:3
25     d4(c)=d3(i+1)-d3(i);
26     c=c+1;
27 end
28 d=[d1(1) d2(1) d3(1) d4(1)];
29 x0=0.638;
30 p=(x0-x(4))/h;
31 y_x=y(4);
32 y_x=y_x+p*(d1(3)+d1(4))/2+p^2*(d2(2))/2; //stirling
    formula
33 printf(' the value at %f by stirling formula is : %f\
    \n\n',x0,y_x);
34 y_x=y(3);
35 p=(x0-x(3))/h;
36 y_x=y_x+p*d1(3)+p*(p-1)*(d2(2)/2);
37 printf(' the value at %f by bessels formula is : %f\
    n\n',x0,y_x);

```

---

### Scilab code Exa 3.12 practical interpolation

```

1 //practical interpolation
2 //example 3.12
3 //page 99
4 clc;clear;close;
5 x=[1.72 1.73 1.74 1.75 1.76 1.77 1.78];
6 y=[0.1790661479 0.1772844100 0.1755204006
    0.1737739435 0.1720448638 0.1703329888
    0.1686381473];
7 h=0.01//interval between values of x

```

```

8 c=1;
9 for i=1:6
10     d1(c)=y(i+1)-y(i);
11     c=c+1;
12 end
13 c=1;
14 for i=1:5
15     d2(c)=d1(i+1)-d1(i);
16     c=c+1
17 end
18 c=1;
19 for i=1:4
20     d3(c)=d2(i+1)-d2(i);
21     c=c+1;
22 end
23 c=1;
24 for i=1:3
25     d4(c)=d3(i+1)-d3(i);
26     c=c+1;
27 end
28 x0=1.7475;
29 y_x=y(3);
30 p=(x0-x(3))/h;
31 y_x=y_x+p*d1(3)+p*(p-1)*((d2(2)+d2(3))/2)/2;
32 printf(' the value at %f by bessels formula is : %0
        .10f\n\n',x0,y_x);
33 y_x=y(4);
34 q=1-p;
35 y_x=q*y(3)+q*(q^2-1)*d2(2)/6+p*y(4)+p*(p^2-1)*d2(2)
        /6;
36 printf(' the value at %f by everrets formula is : %0
        .10f\n\n',x0,y_x);

```

---

**Scilab code Exa 3.13** lagranges interpolation formula

```

1 //example 3.13
2 //lagrange's interpolation formula
3 //page 104
4 clc;clear;close;
5 x=[300 304 305 307];
6 y=[2.4771 2.4829 2.4843 2.4871];
7 x0=301;
8 log_301=0;
9 poly(0, 'x');
10 for i=1:4
11     p=y(i);
12     for j=1:4
13         if i~=j then
14             p=p*((x0-x(j))/(x(i)-x(j)))
15         end
16     end
17     log_301=log_301+p;
18 end
19 disp(log_301, 'log_301=');

```

---

**Scilab code Exa 3.14** lagranges interpolation formula

```

1 //example 3.14
2 //lagrange's interpolation formula
3 //page 105
4 clc;clear;close;
5 y=[4 12 19];
6 x=[1 3 4];
7 y_x=7;
8 Y_X=0;
9 poly(0, 'y');
10 for i=1:3
11     p=x(i);
12     for j=1:3
13         if i~=j then

```



```

14         p=p*((y_x-y(j) )/( y(i)-y(j)))
15         end
16     end
17     Y_X=Y_X+p;
18     end
19 disp(Y_X, 'Y_X=');

```

---

### Scilab code Exa 3.15 lagranges interpolation formula

```

1 //example 3.15
2 //lagrange's interpolation formula
3 //page 105
4 clc;clear;close;
5 x=[2 2.5 3.0];
6 y=[0.69315 0.91629 1.09861];
7 deff('y=l0(x)', 'y=(x-2.5)*(x-3.0)/((-0.5)*(-1.0)')
8 x=poly(0, 'x');
9 disp(l0(x), 'l0(x)=');
10 deff('y=l1(x)', 'y=((x-2.0)*(x-3.0))/((0.5)*(-0.5))')
11 x=poly(0, 'x');
12 disp(l1(x), 'l1(x)=');
13 deff('y=l2(x)', 'y=((x-2.0)*(x-2.5))/((1.0)*(0.5))')
14 x=poly(0, 'x');
15 disp(l2(x), 'l2(x)=');
16 f_x=l0(2.7)*y(1)+l1(2.7)*y(2)+l2(2.7)*y(3);
17 printf(' the calculated value is %f:', f_x);
18 printf('\n\n the error ocured in the value is %0.9f
    ', abs(f_x-log(2.7)))

```

---

### Scilab code Exa 3.16 lagranges interpolation formula

```

1 //example 3.16
2 //lagrange's interpolation formula

```

```

3 //page 104
4 clc;clear;close;
5 x=[0 %pi/4 %pi/2];
6 y=[0 0.70711 1.0];
7 x0=%pi/6;
8 sin_x0=0;
9 poly(0, 'x');
10 for i=1:3
11     p=y(i);
12     for j=1:3
13         if j~=i then
14             p=p*((x0-x(j))/(x(i)-x(j)))
15         end
16     end
17     sin_x0=sin_x0+p;
18 end
19 disp(sin_x0, 'sin_x0=');

```

---

### Scilab code Exa 3.17 lagranges interpolation

```

1 //lagrange's interpolation
2 //example 3.17
3 //page 106
4 clc;clear;close;
5 x=[0 3 4];
6 y=[-12 12 24];
7 //1 appears to be one the roots the polynomial
8 for i=1:3
9     r_x(i)=y(i)/(x(i)-1);
10 end
11 deff('y=l0(x)', 'y=((x-3)*(x-4))/((-3)*(-4))');
12 x=poly(0, 'x');
13 disp(l0(x), 'l0(x)=');
14 deff('y=l1(x)', 'y=((x-0)*(x-4))/((3)*(-1))');
15 x=poly(0, 'x');

```

```

16 disp(l1(x), 'l1(x)=');
17 deff('y=l2(x)', 'y=((x-0)*(x-3))/((4)*(1))')
18 x=poly(0, 'x');
19 disp(l2(x), 'l2(x)=');
20 disp(10(x)*r_x(1)+l1(x)*r_x(2)+l2(x)*r_x(3), 'f_(x)=',
    );
21 disp((x-1)*(10(x)*r_x(1)+l1(x)*r_x(2)+l2(x)*r_x(3))
    ', 'the required polynimial is :')

```

---

**Scilab code Exa 3.18** error in lagranges interpolation formula

```

1 //error in lagrange's interpolation formula
2 //example 3.18
3 //page 107
4 clc;clear;close;
5 x=[2 2.5 3.0];
6 y=[0.69315 0.91629 1.09861];
7 deff('y=l0(x)', 'y=(x-2.5)*(x-3.0)/((-0.5)*(-1.0))')
8 x=poly(0, 'x');
9 disp(l0(x), 'l0(x)=');
10 deff('y=l1(x)', 'y=((x-2.0)*(x-3.0))/((0.5)*(-0.5))')
11 x=poly(0, 'x');
12 disp(l1(x), 'l1(x)=');
13 deff('y=l2(x)', 'y=((x-2.0)*(x-2.5))/((1.0)*(0.5))')
14 x=poly(0, 'x');
15 disp(l2(x), 'l2(x)=');
16 f_x=l0(2.7)*y(1)+l1(2.7)*y(2)+l2(2.7)*y(3);
17 printf(' the calculated value is %f:', f_x);
18 err=abs(f_x-log(2.7));
19 deff('y=R_n(x)', 'y((((x-2)*(x-2.5)*(x-3))/6))');
20 est_err=abs(R_n(2.7)*(2/8))
21 if est_err>err then
22     printf('\n\n the error agrees with the actual
    error ')
23 end

```

---

**Scilab code Exa 3.19** error in lagranges interpolation formula

```
1 //error in lagrenge 's interpolation
2 //example 3.19
3 //page 107
4 clc;clear;close;
5 x=[0 %pi/4 %pi/2];
6 y=[0 0.70711 1.0];
7 deff('y=l0(x)', 'y=((x-0)*(x-%pi/2))/((%pi/4)*(-%pi
    /4))')
8 x=poly(0, 'x');
9 disp(l0(x), 'l0(x)=');
10 deff('y=l1(x)', 'y=((x-0)*(x-%pi/4))/((%pi/2)*(%pi/4)
    )')
11 x=poly(0, 'x');
12 disp(l1(x), 'l1(x)=');
13 f_x=l0(%pi/6)*y(2)+l1(%pi/6)*y(3);
14 err=abs(f_x-sin(%pi/6));
15 deff('y=f(x)', 'y=((x-0)*(x-%pi/4)*(x-%pi/2))/6');
16 if abs(f(%pi/6))>err then
17     printf('\n\n the error agrees with the actual
        error ')
18 end
```

---

**Scilab code Exa 3.21** hermites interpolation formula

```
1 //hermite 's interpolation formula
2 //exammple 3.21
3 //page 110
4 clc;clear;close;
5 x=[2.0 2.5 3.0]
```

```

6 y=[0.69315 0.91629 1.09861]
7 deff('y=f(x)', 'y=log(x)')
8 h=0.0001;
9 for i=1:3
10     y1(i)=(f(x(i)+h)-f(x(i)))/h;
11 end
12 deff('y=l0(x)', 'y=(x-2.5)*(x-3.0)/(-0.5)*(-1.0)')
13 a=poly(0, 'x');
14 disp(10(a), 'l0(x)=');
15 deff('y=l1(x)', 'y=((x-2.0)*(x-3.0))/((0.5)*(-0.5))')
16 a=poly(0, 'x');
17 disp(11(a), 'l1(x)=');
18 deff('y=l2(x)', 'y=((x-2.0)*(x-2.5))/((1.0)*(0.5))')
19 a=poly(0, 'x');
20 disp(12(a), 'l2(x)=');
21 d10=(10(x(1)+h)-10(x(1)))/h;
22 d11=(11(x(2)+h)-11(x(2)))/h;
23 d12=(12(x(3)+h)-12(x(3)))/h;
24 x0=2.7;
25 u0=[1-2*(x0-x(1))*d10]*(10(x0))^2;
26 u1=[1-2*(x0-x(2))*d11]*(11(x0))^2;
27 u2=[1-2*(x0-x(3))*d12]*(12(x0))^2;
28 v0=(x0-x(1))*10(x0)^2;
29 v1=(x0-x(2))*11(x0)^2;
30 v2=(x0-x(3))*12(x0)^2;
31 H=u0*y(1)+u1*y(2)+u2*y(3)+v0*y1(1)+v1*y1(2)+v2*y1(3)
    ;
32 printf(' the approximate value of ln(%0.2f) is %0.6f
    : ', x0, H);

```

---

**Scilab code Exa 3.22** newtons general interpolation formula

```

1 //newton's general interpolation formula
2 //example 3.22
3 //page 114

```

```

4  clc;clear;close;
5  x=[300 304 305 307];
6  y=[2.4771 2.4829 2.4843 2.4871];
7  for i=1:3
8      d1(i)=(y(i+1)-y(i))/(x(i+1)-x(i));
9  end
10 for i=1:2
11     d2(i)=(d1(i+1)-d1(i))/(x(i+2)-x(i));
12 end
13 x0=301;
14 log301=y(1)+(x0-x(1))*d1(1)+(x0-x(2))*d2(1);
15 printf(' value of log(%d) is :%0.4f',x0,log301);

```

---

**Scilab code Exa 3.23** newtons divided formula

```

1  //example 3.22
2  //newton's divided formula
3  //page 114
4  clc;clear;close
5  x=[-1 0 3 6 7];
6  y=[3 -6 39 822 1611];
7  for i=1:4
8      d1(i)=(y(i+1)-y(i))/(x(i+1)-x(i));
9  end
10 for i=1:3
11     d2(i)=(d1(i+1)-d1(i))/(x(i+2)-x(i));
12 end
13 for i=1:2
14     d3(i)=(d2(i+1)-d2(i))/(x(i+3)-x(i));
15 end
16 for i=1:1
17     d4(i)=(d3(i+1)-d3(i))/(x(i+4)-x(i));
18 end
19 X=poly(0,'X')
20 f_x=y(1)+(X-x(1))*(d1(1))+(X-x(2))*(X-x(1))*d2(1)+(X

```

```

    -x(1))*(X-x(2))*(X-x(3))*d3(1)+(X-x(1))*(X-x(2))
    *(X-x(3))*(X-x(4))*d4(1)
21 disp(f_x, 'the polynomial equation is =')

```

---

**Scilab code Exa 3.24** interpolation by iteration

```

1 //interpolation by iteration
2 //example 3.24
3 //page 116
4 clc;clear;close;
5 x=[300 304 305 307];
6 y=[2.4771 2.4829 2.4843 2.4871];
7 x0=301;
8 for i=1:3
9     d=determ([y(i),(x(i)-x0);y(i+1),(x(i+1)-x0)])
10    d1(i)=d/(x(i+1)-x(i));
11 end
12 for i=1:2
13     d=determ([d1(i),(x(i+1)-x0);d1(i+1),(x(i+2)-x0)
14    ])
15     d2(i)=d/(x(i+2)-x(i+1));
16 end
17 for i=1:1
18     d=determ([d2(i),(x(i+2)-x0);d2(i+1),(x(i+3)-x0)
19    ])
20     d3(i)=d/(x(i+3)-x(i+2));
21 end
22 printf(' the value of log(%d) is : %f',x0,d3(1))

```

---

**Scilab code Exa 3.25** inverse intrpolation

```

1 //inverse intrpolation
2 //example 3.25

```

```

3 //page 118
4 clc;clear;close;
5 x=[2 3 4 5];
6 y=[8 27 64 125];
7 for i=1:3
8     d1(i)=y(i+1)-y(i);
9 end
10 for i=1:2
11     d2(i)=d1(i+1)-d1(i);
12 end
13 for i=1:1
14     d3(i)=d2(i+1)-d2(i);
15 end
16 yu=10; //square rooot of 10
17 y0=y(1);
18 d=[d1(1) d2(1) d3(1)];
19 u1=(yu-y0)/d1(1);
20 u2=((yu-y0-u1*(u1-1)*d2(1)/2)/d1(1));
21 u3=(yu-y0-u2*(u2-1)*d2(1)/2-u2*(u2-1)*(u2-2)*d3(1)
    /6)/d1(1);
22 u4=(yu-y0-u3*(u3-1)*d2(1)/2-u3*(u3-1)*(u3-2)*d3(1)
    /6)/d1(1);
23 u5=(yu-y0-u4*(u4-1)*d2(1)/2-u4*(u4-1)*(u4-2)*d3(1)
    /6)/d1(1);
24 printf(' %f \n %f \n %f \n %f \n %f \n ',u1,u2,u3,u4
    ,u5);
25 printf(' the approximate square root of %d is: %0.3f
    ',yu,x(1)+u5)

```

---

### Scilab code Exa 3.26 double interpolation

```

1 //double interpolation
2 //example 3.26
3 //page 119
4 clc;clear;close;

```



```

5 y=[0 1 2 3 4];
6 x=[0 1 4 9 16;2 3 6 11 18;6 7 10 15 22;12 13 16 21
    28;18 19 22 27 34];
7 printf(' y\t\n');
8 for i=1:5
9     printf('\n%d',y(i));
10 end
11 printf('\n\n
    _____x
    _____\n');
12 printf('0\t    1\t    2\t    3\t    4\t\n');
13 printf('
    _____
    n');
14 for i=1:5
15     for j=1:5
16         printf('%d\t',x(i,j));
17     end
18     printf('\n');
19 end
20 //for x=2.5;
21 for i=1:5
22     z(i)=(x(i,3)+x(i,4))/2;
23 end
24 //y=1.5;
25 Z=(z(2)+z(3))/2;
26 printf(' the interpolated value when x=2.5 and y=1.5
    is : %f',Z);
    _____

```

# Chapter 4

## least squares and fourier transform

Scilab code Exa 4.1 least square curve fitting procedure

```
1 //example 4.1
2 //least square curve fitting procedure
3 //page 128
4 clc;clear;close;
5 x=[1 2 3 4 5];
6 y=[0.6 2.4 3.5 4.8 5.7];
7 for i=1:5
8     x_2(i)=x(i)^2;
9     x_y(i)=x(i)*y(i);
10 end
11 S_x=0,S_y=0,S_x2=0,S_xy=0,S1=0,S2=0;
12 for i=1:5
13     S_x=S_x+x(i);
14     S_y=S_y+y(i);
15     S_x2=S_x2+x_2(i);
16     S_xy=S_xy+x_y(i);
17 end
18 a1=(5*S_xy-S_x*S_y)/(5*S_x2-S_x^2);
19 a0=S_y/5-a1*S_x/5;
```

```

20 printf('x\t      y\t      x^2\t      x*y\t      (y-
      avg(S_y))\t(y-a0-a1x)^2\n\n');
21 for i=1:5
22 printf('%d\t      %0.2f\t      %d\t      %0.2f\t
      %0.2f\t      %.4f\t\n',x(i),y(i),
      x_2(i),x_y(i),(y(i)-S_y/5)^2,(y(i)-a0-a1*x(i))^2)
      ;
23 S1=S1+(y(i)-S_y/5)^2;
24 S2=S2+(y(i)-a0-a1*x(i))^2;
25 end
26 printf('
      n\n');
27 printf('%d\t      %0.2f\t      %d\t      %0.2f\t
      %0.2f\t      %0.4f\t\n\n',S_x,S_y,
      S_x2,S_xy,S1,S2);
28 cc=sqrt((S1-S2)/S1); //correlation coefficient
29 printf('the correlation coefficient is:%0.4f',cc);

```

---

#### Scilab code Exa 4.2 least square curve fitting procedure

```

1 //example 4.2
2 //least square curve fitting procedure
3 //page 129
4 clc;clear;close;
5 x=[0 2 5 7];
6 y=[-1 5 12 20];
7 for i=1:4
8     x_2(i)=x(i)^2;
9     xy(i)=x(i)*y(i);
10 end
11 printf('x\t      y\t      x^2\t      xy\t      \n\n');
12 S_x=0,S_y=0,S_x2=0,S_xy=0;
13 for i=1:4
14 printf('%d\t      %d\t      %d\t      %d\t\n',x(i),y(i)

```

```

    ,x_2(i),xy(i));
15 S_x=S_x+x(i);
16 S_y=S_y+y(i);
17 S_x2=S_x2+x_2(i);
18 S_xy=S_xy+xy(i);
19 end
20 printf(' %d\t\t\t\t\t %d\t\t\t\t\t %d\t\t\t\t\t %d\t\t\t\t\t \n',S_x,S_y,
    S_x2,S_xy);
21 A=[4,S_x;S_x,S_x2];
22 B=[S_y;S_xy];
23 C=(A)^-1*B;
24 printf('Best straight line fit Y=%0.4f+x(%0.4f)',C
    (1,1),C(2,1));

```

---

### Scilab code Exa 4.3 least square curve fitting procedure

```

1 //example 4.3
2 //least square curve fitting procedure
3 //page 130
4 clc;clear;close;
5 x=[0 1 2 4 6];
6 y=[0 1 3 2 8];
7 z=[2 4 3 16 8];
8 for i=1:5
9     x2(i)=x(i)^2;
10    y2(i)=y(i)^2;
11    z2(i)=z(i)^2;
12    xy(i)=x(i)*y(i);
13    zx(i)=z(i)*x(i);
14    yz(i)=y(i)*z(i);
15 end
16 S_x=0,S_y=0,S_z=0,S_x2=0,S_y2=0,S_z2=0,S_xy=0,S_zx
    =0,S_yz=0;
17 for i=1:5
18    S_x=S_x+x(i);

```

```

19     S_y=S_y+y(i);
20     S_z=S_z+z(i);
21     S_x2=S_x2+x2(i);
22     S_y2=S_y2+y2(i);
23     S_z2=S_z2+z2(i);
24     S_xy=S_xy+xy(i);
25     S_zx=S_zx+zx(i);
26     S_yz=S_yz+yz(i);
27 end
28 printf('x\t      y\t      z\t      x^2\t      xy\t
          zx\t      y^2\t      yz\n\n');
29 for i=1:5
30     printf('%d\t      %d\t      %d\t      %d\t      %d\t
          %d\t      %d\t      %d\n',x(i),y(i),z(i),x2(i),
          xy(i),zx(i),y2(i),yz(i));
31     end
32 printf('
          n\n');
33 printf('%d\t      %d\t      %d\t      %d\t      %d\t      %d
          \t      %d\t      %d\n\n',S_x,S_y,S_z,S_x2,S_xy,S_zx
          ,S_y2,S_yz);
34 A=[5,13,14;13,57,63;14,63,78];
35 B=[33;122;109];
36 C=A^-1*B;
37 printf('solution of above equation is:a=%d b=%d c=%d
          ',C(1,1),C(2,1),C(3,1));

```

---

#### Scilab code Exa 4.4 linearization of non linear law

```

1 //example 4.4
2 //linearization of non-linear law
3 //page 131
4 clc;clear;close;
5 x=[1 3 5 7 9];

```

```

6 y=[2.473 6.722 18.274 49.673 135.026];
7 for i=1:5
8     Y(i)=log(y(i));
9     x2(i)=x(i)^2;
10    xy(i)=x(i)*Y(i);
11 end
12 S_x=0,S_y=0,S_x2=0,S_xy=0;
13 printf('X\t      Y=lny\t      X^2\t      XY\n\n');
14 for i=1:5
15     printf('%d\t      %0.3f\t      %d\t      %0.3f\n',x(i),
16           Y(i),x2(i),xy(i));
17     S_x=S_x+x(i);
18     S_y=S_y+Y(i);
19     S_x2=S_x2+x2(i);
20     S_xy=S_xy+xy(i);
21 end
22 printf('
n\n')
23 printf('%d\t      %0.3f\t      %d\t      %0.3f\t\n\n',S_x,
24       S_y,S_x2,S_xy);
25 A1=((S_x/5)*S_xy-S_x*S_y)/((S_x/5)*S_x2-S_x^2);
26 A0=(S_y/5)-A1*(S_x/5);
27 a=exp(A0);
28 printf('y=%0.3fexp(%0.2fx)',a,A1);

```

---

#### Scilab code Exa 4.5 linearization of non linear law

```

1 //example 4.5
2 //linearization of non-linear law
3 //page 131
4 clc;clear;close;
5 x=[3 5 8 12];
6 y=[7.148 10.231 13.509 16.434];
7 for i=1:4

```

```

8      X(i)=1/x(i);
9      Y(i)=1/y(i);
10     X2(i)=X(i)^2;
11     XY(i)=X(i)*Y(i);
12 end
13 S_X=0,S_Y=0,S_X2=0,S_XY=0;
14 printf('X\t      Y\t      X^2\t      XY\t\n\n');
15 for i=1:4
16 printf('%0.3f\t      %0.3f\t      %0.3f\t      %0.3f\t\n',X(
      i),Y(i),X2(i),XY(i));
17 S_X=S_X+X(i);
18 S_Y=S_Y+Y(i);
19 S_X2=S_X2+X2(i);
20 S_XY=S_XY+XY(i);
21 end
22 printf('
      n\n');
23 printf('%0.3f\t      %0.3f\t      %0.3f\t      %0.3f\n\n',
      S_X,S_Y,S_X2,S_XY);
24 A1=(4*S_XY-S_X*S_Y)/(4*S_X2-S_X^2);
25 Avg_X=S_X/4;
26 Avg_Y=S_Y/4;
27 A0=Avg_Y-A1*Avg_X;
28 printf('y=x/(%f+%f*x)',A1,A0);

```

---

#### Scilab code Exa 4.6 curve fitting by polynomial

```

1 //example 4.6
2 //curve fitting by polynomial
3 //page 134
4 clc;clear;close;
5 x=[0 1 2];
6 y=[1 6 17];
7 for i=1:3

```

```

8     x2(i)=x(i)^2;
9     x3(i)=x(i)^3;
10    x4(i)=x(i)^4;
11    xy(i)=x(i)*y(i);
12    x2y(i)=x2(i)*y(i);
13  end
14  printf('x\t  y\t  x^2\t  x^3\t  x^4\t  x*y\t
        x^2*y\t\n\n');
15  S_x=0,S_y=0,S_x2=0,S_x3=0,S_x4=0,S_xy=0,S_x2y=0;
16  for i=1:3
17    printf('%d\t  %d\t  %d\t  %d\t  %d\t  %d\t
        %d\n',x(i),y(i),x2(i),x3(i),x4(i),xy(i),x2y
        (i));
18    S_x=S_x+x(i);
19    S_y=S_y+y(i);
20    S_x2=S_x2+x2(i);
21    S_x3=S_x3+x3(i);
22    S_x4=S_x4+x4(i);
23    S_xy=S_xy+xy(i);
24    S_x2y=S_x2y+x2y(i);
25  end
26  printf('
        n\n');
27  printf('%d\t  %d\t  %d\t  %d\t  %d\t  %d\t  %d
        \n',S_x,S_y,S_x2,S_x3,S_x4,S_xy,S_x2y);
28  A=[3,S_x,S_x2;S_x,S_x2,S_x3;S_x2,S_x3,S_x4];
29  B=[S_y;S_xy;S_x2y];
30  C=A^-1*B;
31  printf('a=%d  b=%d  c=%d \n\n',C(1,1),C(2,1),C(3,1))
        ;
32  printf('exact polynomial :%d + %d*x +%d*x^2',C(1,1),
        C(2,1),C(3,1))

```

---

Scilab code Exa 4.7 curve fitting by polynomial



```

1 //example 4.7
2 //curve fitting by polynomial
3 //page 134
4 clc;clear;close;
5 x=[1 3 4 6];
6 y=[0.63 2.05 4.08 10.78];
7 for i=1:4
8     x2(i)=x(i)^2;
9     x3(i)=x(i)^3;
10    x4(i)=x(i)^4;
11    xy(i)=x(i)*y(i);
12    x2y(i)=x2(i)*y(i);
13 end
14 printf('x\t y\t x^2\t x^3\t x^4\t x*y\t
x^2*y\t\n\n');
15 S_x=0,S_y=0,S_x2=0,S_x3=0,S_x4=0,S_xy=0,S_x2y=0;
16 for i=1:4
17     printf('%d\t %0.3f\t %d\t %d\t %d\t %0
.3f\t %d\n',x(i),y(i),x2(i),x3(i),x4(i),xy(
i),x2y(i));
18     S_x=S_x+x(i);
19     S_y=S_y+y(i);
20     S_x2=S_x2+x2(i);
21     S_x3=S_x3+x3(i);
22     S_x4=S_x4+x4(i);
23     S_xy=S_xy+xy(i);
24     S_x2y=S_x2y+x2y(i);
25 end
26 printf('
n\n');
27 printf('%d\t %0.3f\t %d\t %d\t %d\t %0.3f\
t %0.3f\n',S_x,S_y,S_x2,S_x3,S_x4,S_xy,S_x2y);
28 A=[4,S_x,S_x2;S_x,S_x2,S_x3;S_x2,S_x3,S_x4];
29 B=[S_y;S_xy;S_x2y];
30 C=A^-1*B;
31 printf('a=%0.2f b=%0.2f c=%0.2f \n\n',C(1,1),C
(2,1),C(3,1));

```

```
32 printf('exact polynomial :%0.2 f + %0.2 f*x +%0.2 f*x^2
    ',C(1,1),C(2,1),C(3,1))
```

---

#### Scilab code Exa 4.8 curve fitting by sum of exponentials

```
1 //curve fitting by sum of exponentials
2 //example 4.8
3 //page 137
4 clc;clear;close;
5 x=[1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1. 1.9];
6 y=[1.54 1.67 1.81 1.97 2.15 2.35 2.58 2.83 3.11];
7 s1=y(1)+y(5)-2*y(3);
8 h=x(2)-x(1);
9 I1=0;
10 for i=1:3
11     if i==1|i==3 then
12         I1=I1+y(i)
13
14     elseif (modulo(i,2))==0 then
15         I1=I1+4*y(i)
16
17     elseif (modulo(i,2))~=0 then
18         I1=I1+2*y(i)
19     end
20 end
21 I1=(I1*h)/3
22
23 I2=0;
24 for i=3:5
25     if i==3|i==5 then
26         I2=I2+y(i)
27
28     elseif (modulo(i,2))==0 then
29         I2=I2+4*y(i)
30
```

```

31     elseif (modulo(i,2))~=0 then
32         I2=I2+2*y(i)
33         end
34     end
35     I2=(I2*h)/3;
36     for i=1:5
37         y1(i)=(1.0-x(i))*y(i);
38     end
39     for i=5:9
40         y2(i)=(1.4-x(i))*y(i);
41     end
42     I3=0;
43     for i=1:3
44         if i==1|i==3 then
45             I3=I3+y1(i)
46         elseif (modulo(i,2))==0 then
47             I3=I3+4*y1(i)
48             elseif (modulo(i,2))~=0 then
49                 I3=I3+2*y1(i)
50             end
51         end
52         I3=(I3*h)/3
53     I4=0;
54     for i=3:5
55         if i==3|i==5 then
56             I4=I4+y2(i)
57
58         elseif (modulo(i,2))==0 then
59             I4=I4+4*y2(i)
60
61         elseif (modulo(i,2))~=0 then
62             I4=I4+2*y2(i)
63         end
64     end
65     I4=(I4*h)/3;
66     s2=y(5)+y(9)-2*y(7);
67     I5=0;
68     for i=5:7

```

```

69     if i==5|i==7 then
70         I5=I5+y(i)
71     elseif (modulo(i,2))==0 then
72         I5=I5+4*y(i)
73
74     elseif (modulo(i,2))~=0 then
75         I5=I5+2*y(i)
76         end
77     end
78     I5=(I5*h)/3;
79 I6=0;
80 for i=7:9
81     if i==7|i==9 then
82         I6=I6+y(i)
83
84     elseif (modulo(i,2))==0 then
85         I6=I6+4*y(i)
86
87     elseif (modulo(i,2))~=0 then
88         I6=I6+2*y(i)
89         end
90     end
91 I6=(I6*h)/3;
92 I7=0;
93 for i=5:7
94     if i==5|i==7 then
95         I7=I7+y2(i)
96
97     elseif (modulo(i,2))==0 then
98         I7=I7+4*y2(i)
99
100    elseif (modulo(i,2))~=0 then
101        I7=I7+2*y2(i)
102        end
103    end
104    I7=(I7*h)/3;
105 I8=0;
106 for i=7:9

```

```

107     if i==7|i==9 then
108         I8=I8+y2(i)
109
110     elseif (modulo(i,2))==0 then
111         I8=I8+4*y2(i)
112
113     elseif (modulo(i,2))~=0 then
114         I8=I8+2*y2(i)
115     end
116 end
117 I8=(I8*h)/3;
118 A=[1.81 2.180;2.88 3.104];
119 C=[2.10;3.00];
120 Z=A^-1*C
121 X=poly(0,'X');
122 y=X^2-Z(1,1)*X-Z(2,1);
123 R=roots(y)
124 printf(' the unknown value of equation is %1.0f    %1
        .0f ',R(1,1),R(2,1));

```

---

#### Scilab code Exa 4.9 linear weighted least approx

```

1 //linear weighted least approx
2 //example 4.9
3 //page 139
4 clc;clear;close;
5 x=[0 2 5 7];
6 y=[-1 5 12 20];
7 w=10;//given weight 10;
8 W=[1 1 10 1];
9 for i=1:4
10     Wx(i)=W(i)*x(i);
11     Wx2(i)=W(i)*x(i)^2;
12     Wx3(i)=W(i)*x(i)^3;
13     Wy(i)=W(i)*y(i);

```

```

14     Wxy(i)=W(i)*x(i)*y(i);
15     end
16 S_x=0,S_y=0,S_W=0,S_Wx=0,S_Wx2=0,S_Wy=0,S_Wxy=0;
17 for i=1:4
18     S_x=S_x+x(i)
19     S_y=S_y+y(i)
20     S_W=S_W+W(i)
21     S_Wx=S_Wx+Wx(i)
22     S_Wx2=S_Wx2+Wx2(i)
23     S_Wy=S_Wy+Wy(i)
24     S_Wxy=S_Wxy+Wxy(i)
25 end
26 A=[S_W,S_Wx;S_Wx,S_Wx2];
27 C=[S_Wy;S_Wxy];
28 printf('x\t    y\t    W\t    Wx\t    Wx^2\t    Wy\t    Wxy
\t\n\n');
29 for i=1:4
30     printf('%d\t    %d\t    %d\t    %d\t    %d\t    %d\t
%d\t\n',x(i),y(i),W(i),Wx(i),Wx2(i),Wy(i)
,Wxy(i))
31 end
32 printf('
n\n');
33 printf('%d\t    %d\t    %d\t    %d\t    %d\t    %d\t
%d\t\n',S_x,S_y,S_W,S_Wx,S_Wx2,S_Wy,S_Wxy);
34 X=A^-1*C;
35 printf('\n\nthe equation is y=%f+%fx',X(1,1),X(2,1))

```

---

**Scilab code Exa 4.10** linear weighted least approx

```

1 //linear weighted least approx
2 //example 4.10
3 //page 139
4 clc;clear;close;

```

```

5 x=[0 2 5 7];
6 y=[-1 5 12 20];
7 w=100; //given weight 100;
8 W=[1 1 100 1];
9 for i=1:4
10     Wx(i)=W(i)*x(i);
11     Wx2(i)=W(i)*x(i)^2;
12     Wx3(i)=W(i)*x(i)^3;
13     Wy(i)=W(i)*y(i);
14     Wxy(i)=W(i)*x(i)*y(i);
15 end
16 S_x=0, S_y=0, S_W=0, S_Wx=0, S_Wx2=0, S_Wy=0, S_Wxy=0;
17 for i=1:4
18     S_x=S_x+x(i)
19     S_y=S_y+y(i)
20     S_W=S_W+W(i)
21     S_Wx=S_Wx+Wx(i)
22     S_Wx2=S_Wx2+Wx2(i)
23     S_Wy=S_Wy+Wy(i)
24     S_Wxy=S_Wxy+Wxy(i)
25 end
26 A=[S_W, S_Wx; S_Wx, S_Wx2];
27 C=[S_Wy; S_Wxy];
28 printf('x\t y\t W\t Wx\t Wx^2\t Wy\t Wxy
\t\n\n');
29 for i=1:4
30     printf('%d\t %d\t %d\t %d\t %d\t %d\t
%d\t\n', x(i), y(i), W(i), Wx(i), Wx2(i), Wy(i)
, Wxy(i))
31 end
32 printf('
n\n');
33 printf('%d\t %d\t %d\t %d\t %d\t %d\t
%d\t\n', S_x, S_y, S_W, S_Wx, S_Wx2, S_Wy, S_Wxy);
34 X=A^-1*C;
35 printf('\n\nthe equation is y=%f+%fx', X(1,1), X(2,1))
36 printf('\n\nthe value of y(5) is %f', X(1,1)+X(2,1)

```

\*5)

---

**Scilab code Exa 4.11** least square for quadratic equations

```
1 //least square for quadratic equations
2 //example 4.11
3 //page 141
4 clc;clear;close;
5 I1=integrate('1','x',0,%pi/2);
6 I2=integrate('x','x',0,%pi/2);
7 I3=integrate('x^2','x',0,%pi/2);
8 I4=integrate('x^3','x',0,%pi/2);
9 I5=integrate('x^4','x',0,%pi/2);
10 I6=integrate('sin(x)','x',0,%pi/2);
11 I7=integrate('x*sin(x)','x',0,%pi/2);
12 I8=integrate('x^2*sin(x)','x',0,%pi/2);
13 printf('the equations are:\n\n');
14 A=[I1,I2,I3;I2,I3,I4;I3,I4,I5];
15 C=[I6;I7;I8];
16 X=A^-1*C;
17 printf(' the quadratic equation is of the form %f+
    %fx+%fx^2 ',X(1,1),X(2,1),X(3,1));
18 //value of sin pi/4
19 y=X(1,1)+X(2,1)*%pi/4+X(3,1)*(%pi/4)^2
20 printf( '\n\nsin(pi/4)=%0.9f',y)
21 printf('\n\nerror in the preecing solution %0.9f',
    abs(y-sin(%pi/4)))
```

---

**Scilab code Exa 4.20** cooley Tukey method

```
1 //cooley-Tukey method
2 //example 4.20
3 //page 168
```



```

4  clc; clear; close;
5  f=[1,2,3,4,4,3,2,1];
6  F1(1,1)=f(1)+f(5);
7  F1(1,2)=f(1)-f(5);
8  F1(2,1)=f(3)+f(7);
9  F1(2,2)=f(3)-f(7);
10 F1(3,1)=f(2)+f(6);
11 F1(3,2)=f(2)-f(6);
12 F1(4,1)=f(4)+f(8);
13 F1(4,2)=f(4)-f(8);
14 printf('the solutions after first key equation\n\n')
15 disp(F1);
16 F2(1,1)=F1(1,1)+F1(2,1);
17 F2(1,2)=F1(1,1)+F1(2,1);
18 F2(2,1)=F1(1,2)+%i*F1(3,2);
19 F2(2,2)=F1(3,2)-%i*F1(4,2);
20 F2(3,1)=F1(1,1)-F1(2,1);
21 F2(3,2)=F1(1,1)-F1(2,1);
22 F2(4,1)=F1(1,2)+%i*F1(2,2);
23 F2(4,2)=F1(3,2)-%i*F1(1,2);
24 printf('the solutions after second key equation\n\n'
    )
25 disp(F2);
26
27 W=[1,(1-%i)/sqrt(2),-%i,-(1+%i)/sqrt(2),-1,-(1-%i)/
    sqrt(2),%i,(1+%i)/sqrt(2)];
28 F3(1)=F2(1,1)+F2(1,2);
29 F3(2)=F2(2,1)+W(2)*F2(2,2);
30 F3(3)=F2(3,1)+F2(3,2);
31 F3(4)=F2(4,1)+W(4)*F2(4,2);
32 F3(5)=F2(3,1)+F2(3,2);
33 F3(6)=F2(2,1)+W(6)*F2(2,2);
34 F3(7)=F2(3,1)+F2(3,2);
35 F3(8)=F2(4,1)+W(8)*F2(4,2);
36 printf('the solutions after third key equation\n\n')
37 disp(F3);

```

---

# Chapter 5

## spline functions

Scilab code Exa 5.1 linear splines

```
1 //linear splines
2 //example 5.1
3 //page 182
4 clc;clear;close;
5 X=[1 2 3];
6 y=[-8 -1 18];
7 m1=(y(2)-y(1))/(X(2)-X(1));
8 deff('y1=s1(x)', 'y1=y(1)+(x-X(1))*m1');
9 m2=(y(3)-y(2))/(X(3)-X(2));
10 deff('y2=s2(x)', 'y2=y(2)+(x-X(2))*m2');
11 a=poly(0, 'x');
12 disp(s1(a));
13 disp(s2(a));
14 printf(' the value of function at 2.5 is %0.2f: ',s2
      (2.5));
```

---

Scilab code Exa 5.2 quadratic splines

```

1 //quadratic splines
2 //example 5.2
3 //page 18
4 clc;clear;close;
5 X=[1 2 3];
6 y=[-8 -1 18];
7 h=X(2)-X(1);
8 m1=(y(2)-y(1))/(X(2)-X(1));
9 m2=(2*(y(2)-y(1)))/h-m1;
10 m3=(2*(y(3)-y(2)))/h-m2;
11 deff('y2=s2(x)', 'y2=(-(X(3)-x)^2*m1)/2+((x-X(2))^2*
      m3)/2+y(2)+m2/2');
12 a=poly(0, 'x');
13 disp(s2(a));
14 printf('the value of function is 2.5: %0.2f',s2(2.5)
      );
15 x=2.0;
16 h=0.01;
17 deff('y21=s21(x,h)', 'y21=(s2(x+h)-s2(x))/h');
18 d1=s21(x,h);
19 printf('\n\nthe first derivative at 2.0 :%0.2f',d1);

```

---

### Scilab code Exa 5.3 cubic splines

```

1 //cubic splines
2 //example 5.3
3 //page 188
4 clc;clear;close;
5 X=[1 2 3];
6 y=[-8 -1 18];
7 M1=0,M2=8,M3=0;
8 h=1;
9 deff('y=s1(x)', 'y=3*(x-1)^3-8*(2-x)-4*(x-1)');
10 deff('y=s2(x)', 'y=3*(3-x)^3+22*x-48');
11 h=0.0001;n=2.0;

```

```

12 D=(s2(n+h)-s2(n))/h;
13 a=poly(0,'x');
14 disp(s1(a),' s1(x)=');
15 disp(s2(a),' s2(x)=');
16 disp(s2(2.5),' y(2.5)=');
17 disp(D,' y1(2.0)=');

```

---

### Scilab code Exa 5.4 cubic splines

```

1 //cubic spline
2 //example 5.4
3 //page 189
4 clc;clear;close;
5 x=[0 %pi/2 %pi]
6 y=[0 1 0]
7 h=x(2)-x(1)
8 M0=0;M2=0;
9 M1=((6*(y(1)-2*y(2)+y(3))/h^2)-M0-M2)/4;
10 X=%pi/6;
11 s1=((x(2)-X)^3)*(M0/6)+((X-x(1))^3)*M1/6+(y(1)-(h
    ^2)*M0/6)*(x(2)-X)+(y(2)-(h^2)*M1/6)*(X-x(1)))/h;
12 x=[0 %pi/4 %pi/2 3*%pi/4 %pi];
13 y=[0 1.414 1 1.414];
14 M0=0,M4=0;
15 A=[4 1 0;1 4 1;0 1 4];//calculating value of M1 M2
    M3 by matrix method
16 C=[-4.029;-5.699;-4.029];
17 B=A^-1*C
18 printf('M0=%f\t M1=%f\t M2=%f\t M3=%f\t M4=
    %f\t\n\n',M0,B(1,1),B(2,1),B(3,1),M4);
19 h=%pi/4;
20 X=%pi/6;
21 s1=[-0.12408*X^3+0.7836*X]/h;
22 printf(' the value of sin(pi/6) is:%f',s1)

```

---

### Scilab code Exa 5.5 cubic splines

```
1 //cubic spline
2 //example 5.5
3 //page 191
4 clc;clear;close;
5 x=[1 2 3];
6 y=[6 18 42];
7 m0=40;
8 m1=(3*(y(3)-y(1))-m0)/4;
9 X=poly(0, 'X');
10 s1=m0*((x(2)-X)^2)*(X-x(1))-m1*((X-x(1))^2)*(x(2)-X)
    +y(1)*((x(2)-X)^2)*[2*(X-x(1))+1]+y(2)*((X-x(1))
    ^2)*[2*(x(2)-X)+1];
11 disp(s1, 's1=');
```

---

### Scilab code Exa 5.7 surface fitting by cubic spline

```
1 //surface fitting by cubic spline
2 //example 5.7
3 //page 195
4 clc;clear;close;
5 z=[1 2 9;2 3 10;9 10 17];
6 deff('y=L0(x)', 'y=x^3/4-5*x/4+1');
7 deff('y=L1(x)', 'y=-x^3/2+3*x/2');
8 deff('y=L2(x)', 'y=x^3/4-x/4');
9 x=0.5;y=0.5;
10 S=0;
11 S=S+L0(x)*(L0(x)*z(1,1)+L1(x)*z(1,2)+L2(x)*z(1,3));
12 S=S+L1(x)*(L0(x)*z(2,1)+L1(x)*z(2,2)+L2(x)*z(2,3));
13 S=S+L2(x)*(L0(x)*z(3,1)+L1(x)*z(3,2)+L2(x)*z(3,3));
14 printf('approximated value of z(0.5 0.5)=%f\n\n',S);
```

```
15 printf(' error in the approximated value : %f',(abs
    (1.25-S)/1.25)*100)
```

---

### Scilab code Exa 5.8 cubic B splines

```
1 //cubic B-splines
2 //example 5.8
3 //page 200
4 clc;clear;close;
5 k=[0 1 2 3 4];
6 pi=[0 0 4 -6 24];
7 x=1;
8 S=0;
9 for i=3:5
10     S=S+((k(i)-x)^3)/(pi(i));
11 end
12 printf(' the cubic splines for x=1 is %f\n\n',S);
13 S=0;
14 x=2;
15 for i=4:5
16     S=S+((k(i)-x)^3)/(pi(i));
17 end
18 printf(' the cubic splines for x=2 is %f\n\n',S);
```

---

### Scilab code Exa 5.9 cubic B spline

```
1 //cubic B-spline
2 //example 5.8
3 //page 201
4 clc;clear;close;
5 k=[0 1 2 3 4];
6 x=1; //for x=1
7 s11=0; s13=0; s14=0;
```

```

8 s24=0;
9 s12=1/(k(3)-k(2));
10 s22=((x-k(1))*s11+(k(3)-x)*s12)/(k(3)-k(1));
11 s23=((x-k(2))*s11+(k(4)-x)*s13)/(k(4)-k(2));
12 s33=((x-k(1))*s22+(k(4)-x)*s23)/(k(4)-k(1));
13 s34=((x-k(2))*s23+(k(5)-x)*s24)/(k(5)-k(2));
14 s44=((x-k(1))*s33+(k(5)-x)*s34)/(k(5)-k(1));
15 printf( 's11=%f\t      s22=%f\t      s23=%f\t      s33=%f
\t      s34=%f\t      s44=%f\n\n',s11 ,s22 ,s23 ,s33 ,s34 ,
s44);
16 x=2; //for x=2;
17 s11=0; s12=0 , s14=0; s22=0;
18 s13=1/(k(4)-k(3));
19 s23=((x-k(2))*s12+(k(4)-x)*s13)/(k(4)-k(2));
20 s24=((x-k(3))*s13+(k(5)-x)*s14)/(k(3)-k(1));
21 s33=((x-k(1))*s22+(k(4)-x)*s23)/(k(4)-k(1));
22 s34=((x-k(2))*s23+(k(5)-x)*s24)/(k(5)-k(2));
23 s44=((x-k(1))*s33+(k(5)-x)*s34)/(k(5)-k(1));
24 printf( 's13=%f\t      s23=%f\t      s24=%f\t      s33=
%f\t      s34=%f\t      s44=%f\n\n',s13 ,s23 ,s24 ,s33 ,
s34 ,s44);

```

---

# Chapter 6

## Numerical Differentiation and Integration

Scilab code Exa 6.1 numerical differentiation by newtons difference formula

```
1 //example 6.1
2 //numerical differentiation by newton's difference
   formula
3 //page 210
4 clc;clear;close
5 x=[1.0 1.2 1.4 1.6 1.8 2.0 2.2];
6 y=[2.7183 3.3201 4.0552 4.9530 6.0496 7.3891
   9.0250];
7 c=1;
8 for i=1:6
9     d1(c)=y(i+1)-y(i);
10    c=c+1;
11 end
12 c=1;
13 for i=1:5
14    d2(c)=d1(i+1)-d1(i);
15    c=c+1;
16 end
```



```

17 c=1;
18 for i=1:4
19     d3(c)=d2(i+1)-d2(i);
20     c=c+1;
21 end
22 c=1;
23 for i=1:3
24     d4(c)=d3(i+1)-d3(i);
25     c=c+1;
26 end
27 c=1;
28 for i=1:2
29     d5(c)=d4(i+1)-d4(i);
30     c=c+1;
31 end
32 c=1;
33 for i=1:1
34     d6(c)=d5(i+1)-d5(i);
35     c=c+1;
36 end
37 x0=1.2//first and second derivative at 1.2
38 h=0.2;
39 f1=((d1(2)-d2(2)/2+d3(2)/3-d4(2)/4+d5(2)/5)/h);
40 printf('the first derivative of fuction at 1.2 is:%f
    \n',f1);
41 f2=(d2(2)-d3(2)+(11*d4(2))/12-(5*d5(2))/6)/h^2;
42 printf('the second derivative of fuction at 1.2 is:
    %f\n',f2);

```

---

**Scilab code Exa 6.2** numerical diffrentiation by newtons difference formula

```

1 //example 6.2
2 //numerical diffrentiation by newton's difference
  formula

```

```

3 //page 211
4 clc;clear;close
5 x=[1.0 1.2 1.4 1.6 1.8 2.0 2.2];
6 y=[2.7183 3.3201 4.0552 4.9530 6.0496 7.3891
    9.0250];
7 c=1;
8 for i=1:6
9     d1(c)=y(i+1)-y(i);
10    c=c+1;
11 end
12 c=1;
13 for i=1:5
14    d2(c)=d1(i+1)-d1(i);
15    c=c+1;
16 end
17 c=1;
18 for i=1:4
19    d3(c)=d2(i+1)-d2(i);
20    c=c+1;
21 end
22 c=1;
23 for i=1:3
24    d4(c)=d3(i+1)-d3(i);
25    c=c+1;
26 end
27 c=1;
28 for i=1:2
29    d5(c)=d4(i+1)-d4(i);
30    c=c+1;
31 end
32 c=1;
33 for i=1:1
34    d6(c)=d5(i+1)-d5(i);
35    c=c+1;
36 end
37 x0=2.2//first and second derivative at 2.2
38 h=0.2;
39 f1=((d1(6)+d2(5)/2+d3(4)/3+d4(3)/4+d5(2)/5)/h);

```

```

40 printf('the first derivative of fuction at 1.2 is:%f
    \n',f1);
41 f2=(d2(5)+d3(4)+(11*d4(3))/12+(5*d5(2))/6)/h^2;
42 printf('the second derivative of fuction at 1.2 is:
    %f\n',f2);
43 x1=2.0; // first derivative also at 2.0
44 f1=((d1(5)+d2(4)/2+d3(3)/3+d4(2)/4+d5(1)/5+d6(1)/6)/
    h);
45 printf('\n \nthe first derivative of function at 1.2
    is:%f\n',f1);

```

---

**Scilab code Exa 6.3** numerical diffrentiation by newtons difference formula

```

1 //example 6.3
2 //numerical diffrentiation by newton's difference
    formula
3 //page 211
4 clc;clear;close
5 x=[1.0 1.2 1.4 1.6 1.8 2.0 2.2];
6 y=[2.7183 3.3201 4.0552 4.9530 6.0496 7.3891
    9.0250];
7 c=1;
8 for i=1:6
9     d1(c)=y(i+1)-y(i);
10    c=c+1;
11 end
12 c=1;
13 for i=1:5
14    d2(c)=d1(i+1)-d1(i);
15    c=c+1;
16 end
17 c=1;
18 for i=1:4
19    d3(c)=d2(i+1)-d2(i);

```

```

20     c=c+1;
21 end
22 c=1;
23 for i=1:3
24     d4(c)=d3(i+1)-d3(i);
25     c=c+1;
26 end
27 c=1;
28 for i=1:2
29     d5(c)=d4(i+1)-d4(i);
30     c=c+1;
31 end
32 c=1;
33 for i=1:1
34     d6(c)=d5(i+1)-d5(i);
35     c=c+1;
36 end
37 x0=1.6 // first and second derivative at 1.6
38 h=0.2;
39 f1=(((d1(3)+d1(4))/2-(d3(2)+d3(3))/4+(d5(1)+d5(2))
      /60))/h
40 printf('the first derivative of function at 1.6 is:
      %f\n',f1);
41 f2=((d2(3)-d4(2)/12)+d6(1)/90)/(h^2);
42 printf('the second derivative of function at 1.6 is:
      %f\n',f2);

```

---

#### Scilab code Exa 6.4 estimation of errors

```

1 //example 6.4
2 //estimation of errors
3 //page 213
4 clc;clear;close
5 x=[1.0 1.2 1.4 1.6 1.8 2.0 2.2];
6 y=[2.7183 3.3201 4.0552 4.9530 6.0496 7.3891

```

```

    9.0250];
7  c=1;
8  for i=1:6
9      d1(c)=y(i+1)-y(i);
10     c=c+1;
11 end
12 c=1;
13 for i=1:5
14     d2(c)=d1(i+1)-d1(i);
15     c=c+1;
16 end
17 c=1;
18 for i=1:4
19     d3(c)=d2(i+1)-d2(i);
20     c=c+1;
21 end
22 c=1;
23 for i=1:3
24     d4(c)=d3(i+1)-d3(i);
25     c=c+1;
26 end
27 c=1;
28 for i=1:2
29     d5(c)=d4(i+1)-d4(i);
30     c=c+1;
31 end
32 c=1;
33 for i=1:1
34     d6(c)=d5(i+1)-d5(i);
35     c=c+1;
36 end
37 x0=1.6//first and second derivative at 1.6
38 h=0.2;
39 f1=((d1(2)-d2(2)/2+d3(2)/3-d4(2)/4+d5(2)/5)/h);
40 printf('the first derivative of fuction at 1.2 is:%f
    \n',f1);
41 f2=(d2(2)-d3(2)+(11*d4(2))/12-(5*d5(2))/6)/h^2;
42 printf('the second derivative of fuction at 1.2 is:

```

```

    %f\n',f2);
43 T_error1=((d3(2)+d3(3))/2)/(6*h); //truncation error
44 e=0.00005; //corrected to 4D values
45 R_error1=(3*e)/(2*h);
46 T_error1=T_error1+R_error1; //total error
47 f11=(d1(3)+d1(4))/(2*h); //using stirling formula
    first derivative
48 f22=d2(3)/(h*h); //second derivative
49 T_error2=d4(2)/(12*h*h);
50 R_error2=(4*e)/(h*h);
51 T_error2=T_error2+R_error2;
52 printf('total error in first derivative is %0.4g:\n
    ',T_error1);
53 printf('total error in second derivative is %0.4g:',
    T_error2);

```

---

#### Scilab code Exa 6.5 cubic spline method

```

1 //cubic spline method
2 //example 6.5
3 //page 214
4 clc;clear;close;
5 x=[0 %pi/2 %pi];
6 y=[0 1 0];
7 M0=0,M2=0;
8 h=%pi/2;
9 M1=(6*(y(1)-2*y(2)+y(3))/(h^2)-M0-M2)/4;
10 deff('y=s1(x)', 'y=2*((-2*3*x^2)/(%pi^2)+3/2)/%pi');
11 S1=s1(%pi/4);
12 disp(S1, 'S1(pi/4)=');
13 deff('y=s2(x)', 'y=(-24*x)/(%pi^3)');
14 S2=s2(%pi/4);
15 disp(S2, 'S2(pi/4)=');

```

---

**Scilab code Exa 6.6** derivative by cubic spline method

```
1 //derivative by cubic spline method
2 //example 6.6
3 //page 216
4 clc;clear;close;
5 x=[-2 -1 2 3];
6 y=[-12 -8 3 5]
7 def('y=f(x)', 'y=x^3/15-3*x^2/20+241*x/60-3.9');
8 def('y=s2(x)', 'y=[((2-x)^3/6)*(14/55)+(x+1)
    ^3/6*(-74/55)]/3+[-8-21/55]*(2-x)/3+[3-(9/6)
    *(-74/55)]*(x+1)/3');
9 h=0.0001;
10 x0=1.0;
11 y1=(s2(x0+h)-s2(x0))/h;
12 printf(' the value y1(%0.2f) is : %f',x0,y1);
```

---

**Scilab code Exa 6.7** maximum and minimum of functions

```
1 //maximum and minimum of functions
2 //example 6.7
3 //page 218
4 clc;clear;close;
5 x=[1.2 1.3 1.4 1.5 1.6];
6 y=[0.9320 0.9636 0.9855 0.9975 0.9996];
7 for i=1:4
8     d1(i)=y(i+1)-y(i);
9 end
10 for i=1:3
11     d2(i)=d1(i+1)-d1(i);
12 end
13 p=(-d1(1)*2/d2(1)+1)/2;
```

```

14 disp(p, 'p=');
15 h=0.1;
16 x0=1.2;
17 X=x0+p*h;
18 printf(' the value of X correct to 2 decimal places
    is : %0.2f',X);
19 Y=y(5)-0.2*d1(4)+(-0.2)*(-0.2+1)*d2(3)/2;
20 disp(Y, 'the value Y=');

```

---

**Scilab code Exa 6.8** trapezoidal method for integration

```

1 //example 6.8
2 //trapezoidal method for integration
3 //page 226
4 clc;clear;close;
5 x=[7.47 7.48 7.49 7.0 7.51 7.52];
6 f_x=[1.93 1.95 1.98 2.01 2.03 2.06];
7 h=x(2)-x(1);
8 l=length(x);
9 area=0;
10 for i=1:l
11     if i==1|i==l then
12         area=area+f_x(i)
13     else
14         area=area+2*f_x(i)
15     end
16 end
17 area=area*(h/2);
18 printf('area bounded by the curve is %f',area);

```

---

**Scilab code Exa 6.9** simpson 1by3 method for integration

```

1 //example 6.8

```



```

2 //simpson 1/3rd method for integration
3 //page 226
4 clc;clear;close;
5 x=[0.00 0.25 0.50 0.75 1.00];
6 y=[1.000 0.9896 0.9589 0.9089 0.8415];
7 y=y^2;
8 h=x(2)-x(1);
9 l=length(x);
10 area=0;
11 for i=1:l
12     if i==1|i==l then
13         area=area+y(i)
14
15     elseif (modulo(i,2))==0 then
16         area=area+4*y(i)
17
18     elseif (modulo(i,2))~=0 then
19         area=area+2*y(i)
20         end
21     end
22 area=area*(h*pi)/3;
23 printf('area bounded by the curve is %f',area);

```

---

**Scilab code Exa 6.10** integration by trapezoidal and simpsons method

```

1 //example 6.10
2 //integration by trapezoidal and simpson's method
3 //page 228
4 clc;clear;close
5 deff('y=f(x)', 'y=1/(1+x)');
6 h=0.5;
7 x=0:h:1;
8 l=length(x);
9 for i=1:l
10     y(i)=f(x(i));

```

```

11 end
12 area=0;//trapezoidal method
13 for i=1:l
14     if i==1|i==l then
15         area=area+y(i)
16     else
17         area=area+2*y(i)
18     end
19 end
20 area=area*(h/2);
21 printf('area bounded by the curve by trapezoidal
        method with h=%f is %f\n \n',h,area);
22 area=0;//simpson 1/3rd rule
23 for i=1:l
24     if i==1|i==l then
25         area=area+y(i)
26
27     elseif (modulo(i,2))==0 then
28         area=area+4*y(i)
29
30     elseif (modulo(i,2))~=0 then
31         area=area+2*y(i)
32     end
33 end
34 area=(area*h)/3;
35 printf('area bounded by the curve by simpson 1/3rd
        method with h=%f is %f\n \n',h,area);
36 h=0.25;
37 x=0:h:1;
38 l=length(x);
39 for i=1:l
40     y(i)=f(x(i));
41 end
42 area=0;//trapezoidal method
43 for i=1:l
44     if i==1|i==l then
45         area=area+y(i)
46     else

```

```

47         area=area+2*y(i)
48     end
49 end
50 area=area*(h/2);
51 printf('area bounded by the curve by trapezoidal
        method with h=%f is %f\n \n',h,area);
52 area=0;//simpson 1/3rd rule
53 for i=1:l
54     if i==1|i==l then
55         area=area+y(i)
56
57     elseif (modulo(i,2))==0 then
58         area=area+4*y(i)
59
60     elseif (modulo(i,2))~=0 then
61         area=area+2*y(i)
62     end
63 end
64 area=(area*h)/3;
65 printf('area bounded by the curve by simpson 1/3rd
        method with h=%f is %f\n \n',h,area);
66 h=0.125;
67 x=0:h:1;
68 l=length(x);
69 for i=1:l
70     y(i)=f(x(i));
71 end
72 area=0;//trapezoidal method
73 for i=1:l
74     if i==1|i==l then
75         area=area+y(i)
76     else
77         area=area+2*y(i)
78     end
79 end
80 area=area*(h/2);
81 printf('area bounded by the curve by trapezoidal
        method with h=%f is %f\n \n',h,area);

```

```

82 area=0;//simpson 1/3rd rule
83 for i=1:l
84     if i==1|i==l then
85         area=area+y(i)
86
87     elseif (modulo(i,2))==0 then
88         area=area+4*y(i)
89
90     elseif (modulo(i,2))~=0 then
91         area=area+2*y(i)
92     end
93 end
94 area=(area*h)/3;
95 printf('area bounded by the curve by simpson 1/3rd
        method with h=%f is %f\n \n',h,area);

```

---

#### Scilab code Exa 6.11 rommbergs method

```

1 //example 6.11
2 //rommberg's method
3 //page 229
4 clc;clear;close;
5 deff('y=f(x)', 'y=1/(1+x)');
6 k=1;
7 h=0.5;
8 x=0:h:1;
9 l=length(x);
10 for i=1:l
11     y(i)=f(x(i));
12 end
13 area=0;//trapezoidal method
14 for i=1:l
15     if i==1|i==l then
16         area=area+y(i)
17     else

```

```

18         area=area+2*y(i)
19     end
20 end
21 area=area*(h/2);
22 I(k)=area;
23 k=k+1;
24 h=0.25;
25 x=0:h:1;
26 l=length(x);
27 for i=1:l
28     y(i)=f(x(i));
29 end
30 area=0;//trapezoidal method
31 for i=1:l
32     if i==1|i==l then
33         area=area+y(i)
34     else
35         area=area+2*y(i)
36     end
37 end
38 area=area*(h/2);
39 I(k)=area;
40 k=k+1;
41 h=0.125;
42 x=0:h:1;
43 l=length(x);
44 for i=1:l
45     y(i)=f(x(i));
46 end
47 area=0;//trapezoidal method
48 for i=1:l
49     if i==1|i==l then
50         area=area+y(i)
51     else
52         area=area+2*y(i)
53     end
54 end
55 area=area*(h/2);

```

```

56 I(k)=area;
57 printf('results obtained with h=0.5 0.25 0.125 is %f
        %f %f\n \n',I(1),I(2),I(3));
58 for i=1:2
59     I1(i)=I(i+1)+(I(i+1)-I(i))/3
60 end
61 for i=1:1
62     T2(i)=I1(i+1)+(I1(i+1)-I1(i))/3
63 end
64 printf('the area is %f',T2(1))

```

---

#### Scilab code Exa 6.12 Trapezoidal and Simpsons rule

```

1 //example 6.12
2 //Trapezoidal and Simpson's rule
3 //page 230
4 clc;clear;close;
5 deff('y=f(x)', 'y=sqrt(1-x^2)');
6 k=10:10:50
7 for i=1:length(k)
8     T_area(i)=0,S_area(i)=0;
9     h=1/k(i);
10    x=0:h:1
11    l=length(x);
12    for j=1:l
13        y(j)=f(x(j));
14    end
15    for j=1:l
16        if j==1|j==l then
17            T_area(i)=T_area(i)+y(j)
18        else
19            T_area(i)=T_area(i)+2*y(j)
20        end
21    end
22 end

```

```

23 T_area(i)=T_area(i)*(h/2);
24 for j=1:l
25     if j==1|j==l then
26         S_area(i)=S_area(i)+y(j)
27
28     elseif (modulo(j,2))==0 then
29         S_area(i)=S_area(i)+4*y(j)
30
31     elseif (modulo(i,2))~=0 then
32         S_area(i)=S_area(i)+2*y(j)
33     end
34 end
35 S_area(i)=S_area(i)*(h)/3;
36 end
37 printf(' no of subintervals      Trapezoidal Rule
        Simpsons Rule\t \n \n')
38 for i=1:length(k)
39     printf(' %0.9g                %0.9g
        %0.9g                \n ',k(i),T_area
        (i),S_area(i));
40
41 end

```

---

### Scilab code Exa 6.13 area using cubic spline method

```

1 //area using cubic spline method
2 //example 6.2
3 //page 230
4 clc;clear;close;
5 x=[0 0.5 1.0];
6 y=[0 1.0 0.0]
7 h=0.5;
8 M0=0,M2=0;
9 M1=(6*(y(3)-2*y(2)+y(1)))/h^2-M0-M2)/4;
10 M=[M0 M1 M2];

```

```

11 I=0;
12 for i=1:2
13     I=I+(h*(y(i)+y(i+1)))/2-((h^3)*(M(i)+M(i+1)))/24)
        ;
14 end
15 printf(' the value of the integrand is : %f',I);

```

---

**Scilab code Exa 6.15** eulers maclaurin formula

```

1 //euler 's maclaurin formula
2 //example 6.15
3 //page 233
4 clc;clear;close;
5 y=[0 1 0];
6 h=%pi/4;
7 I=h*(y(1)+2*y(2)+y(3))/2+(h^2)/12+(h^4)/720;
8 printf(' the value of integrand with h=%f is : %f\n\
        n',h,I)
9 h=%pi/8;
10 y=[0 sin(%pi/8) sin(%pi*2/8) sin(%pi*3/8) sin(%pi
        *4/8)]
11 I=h*(y(1)+2*y(2)+2*y(3)+2*y(4)+y(5))/2+(h^2)/2+(h^2)
        /12+(h^4)/720;
12 printf(' the value of integrand with h=%f is : %f',h
        ,I)

```

---

**Scilab code Exa 6.17** error estimate in evaluation of the integral

```

1 // example 6.17
2 // error estimate in evaluation of the integral
3 // page 236
4 clc;clear;close;
5 def('z=f(a,b)', 'z=cos(a)+4*cos((a+b)/2)+cos(b)')

```



```

6 a=0,b=%pi/2,c=%pi/4;
7 I(1)=(f(a,b)*((b-a)/2)/3)
8 I(2)=(f(a,c)*((c-a)/2)/3)
9 I(3)=(f(c,b)*((b-c)/2)/3)
10 Area=I(2)+I(3);
11 Error_estimate=((I(1)-I(2)-I(3))/15);
12 Actual_area=integrate('cos(x)','x',0,%pi/2);
13 Actual_error=abs(Actual_area-Area);
14 printf('the calculated area obtained is:%f\n',Area)
15 printf('the actual area obtained is:%f\n',
        Actual_area)
16 printf('the actual error obtained is:%f\n',
        Actual_error)

```

---

**Scilab code Exa 6.18** error estimate in evaluation of the integral

```

1 // example 6.18
2 // error estimate in evaluation of the integral
3 // page 237
4 clc;clear;close;
5 deff('z=f(a,b)','z=8+4*sin(a)+4*(8+4*sin((a+b)/2))
        +8+4*sin(b)')
6 a=0,b=%pi/2,c=%pi/4;
7 I(1)=(f(a,b)*((b-a)/2)/3)
8 I(2)=(f(a,c)*((c-a)/2)/3)
9 I(3)=(f(c,b)*((b-c)/2)/3)
10 Area=I(2)+I(3);
11 Error_estimate=((I(1)-I(2)-I(3))/15);
12 Actual_area=integrate('8+4*cos(x)','x',0,%pi/2);
13 Actual_error=abs(Actual_area-Area);
14 printf('the calculated area obtained is:%f\n',Area)
15 printf('the actual area obtained is:%f\n',
        Actual_area)
16 printf('the actual error obtained is:%f\n',
        Actual_error)

```

---

**Scilab code Exa 6.19** gauss formula

```
1 //gauss ' formula
2 //example 6.19
3 //page 242
4 clc;clear;close;
5 u=[-0.86113 -0.33998 0.33998 0.86113];
6 W=[0.34785 0.65214 0.65214 0.34785];
7 I=0;
8 for i=1:4
9     I=I+(u(i)+1)*W(i);
10 end
11 I=I/4;
12 printf(' the value of integrand is : %0.5f',I);
```

---

**Scilab code Exa 6.20** double integration

```
1 //example 6.20
2 //double integration
3 //page 247
4 clc;clear;close;
5 deff('z=f(x,y)', 'z=exp(x+y)');
6 h0=0.5,k0=0.5;
7 h=[0 0.5 1];,k=[0 0.5 1];
8 for i=1:3
9     for j=1:3
10        x(i,j)=f(h(i),k(j));
11    end
12 end
13 T_area=h0*k0*(x(1,1)+4*x(1,2)+4*x(3,2)+6*x(1,3)+x
    (3,3))/4//trapezoidal method
```

```
14 printf('the integration value by trapezoidal method
    is %f\n ',T_area);
15 S_area=h0*k0*((x(1,1)+x(1,3)+x(3,1)+x(3,3)+4*(x(1,2)
    +x(3,2)+x(2,3)+x(2,1))+16*x(2,2))/9
16 printf('the integration value by Simpson method is
    %f',S_area);
```

---

# Chapter 7

## Numerical linear algebra

Scilab code Exa 7.1 inverse of matrix

```
1 //example 7.1
2 //inverse of matrix
3 //page 256
4 clc;clear;close;
5 A=[1,2,3;0,1,2;0,0,1];
6 A_1=1/A//inverse of matrix
7 for i=1:3
8     for j=1:3
9         printf('%d  ',A_1(i,j))
10    end
11    printf('\n')
12 end
```

---

Scilab code Exa 7.2 Factorize by triangulation method

```
1 //example 7.2
2 //Factorize by triangulation method
3 //page 259
```

```

4  clc;clear;close;
5  A=[2,3,1;1,2,3;3,1,2];
6  L(1,2)=0,L(1,3)=0,L(2,3)=0;
7  U(2,1)=0,U(3,1)=0,U(3,2)=0;
8  for i=1:3
9      L(i,i)=1;
10 end
11 for i=1:3
12     U(1,i)=A(1,i);
13 end
14 L(2,1)=1/U(1,1);
15 for i=2:3
16     U(2,i)=A(2,i)-U(1,i)*L(2,1);
17 end
18 L(3,1)=A(3,1)/U(1,1);
19 L(3,2)=(A(3,2)-U(1,2)*L(3,1))/U(2,2);
20 U(3,3)=A(3,3)-U(1,3)*L(3,1)-U(2,3)*L(3,2);
21 printf('The Matrix A in Triangle form\n \n')
22 printf('Matrix L\n');
23 for i=1:3
24     for j=1:3
25         printf('%.2f  ',L(i,j));
26     end
27     printf('\n');
28 end
29 printf('\n \n');
30 printf('Matrix U\n');
31 for i=1:3
32     for j=1:3
33         printf('%.2f  ',U(i,j));
34     end
35     printf('\n');
36 end

```

---

Scilab code Exa 7.3 Vector Norms

```

1 //example 7.3
2 //Vector Norms
3 //page 262
4 clc;clear;close;
5 A=[1,2,3;4,5,6;7,8,9];
6 s=0;
7 for i=1:3
8     for j=1:3
9         s=s+A(j,i);
10    end
11    C(i)=s;
12    s=0;
13 end
14 printf(' ||A||1=%d\n',max(C));
15 for i=1:3
16     for j=1:3
17         s=s+(A(i,j)*A(i,j))
18     end
19 end
20 printf(' ||A|| e=%0.3 f\n',sqrt(s));
21 s=0;
22 for i=1:3
23     for j=1:3
24         s=s+A(i,j);
25     end
26    C(i)=s;
27    s=0;
28 end
29 printf(' ||A||~=%d\n',max(C));

```

---

### Scilab code Exa 7.6 Gauss Jordan

```

1 //example 7.4
2 //Gauss Jordan
3 //page 266

```

```

4  clc;clear;close;
5  A=[2,1,1,10;3,2,3,18;1,4,9,16]; //augmented matrix
6  for i=1:3
7      j=i
8      while (A(i,i)==0&j<=3)
9
10     for k=1:4
11         B(1,k)=A(j+1,k)
12         A(j+1,k)=A(i,k)
13         A(i,k)=B(1,k)
14     end
15     disp(A);
16     j=j+1;
17 end
18 disp(A);
19 for k=4:-1:i
20     A(i,k)=A(i,k)/A(i,i)
21 end
22 disp(A)
23 for k=1:3
24     if(k~=i) then
25         l=A(k,i)/A(i,i)
26         for m=i:4
27             A(k,m)=A(k,m)-l*A(i,m)
28         end
29     end
30 end
31 disp(A)
32 end
33 for i=1:3
34     printf( '\nx(%i )=%g\n',i,A(i,4))
35 end

```

---

Scilab code Exa 7.7 modern gauss jordan method

```

1 //modern gauss jordan method
2 //example 7.7
3 //page 269
4 clc;clear;close;
5 A=[2 1 1;3 2 3;1 4 9];
6 I=eye(3,3);
7 I1=[1;0;0];
8 I2=[0;1;0];
9 I3=[0;0;1];
10 A1=A^-1*I1;
11 A2=A^-1*I2;
12 A3=A^-1*I3;
13 for i=1:3
14     AI(i,1)=A1(i,1)
15 end
16 for i=1:3
17     AI(i,2)=A2(i,1)
18 end
19 for i=1:3
20     AI(i,3)=A3(i,1)
21 end
22 printf('the inverse of the matrix\n')
23 for i=1:3
24     for j=1:3
25         printf('%0.2g      ',AI(i,j))
26     end
27     printf('\n');
28 end

```

---

### Scilab code Exa 7.8 LU decomposition method

```

1 //LU decomposition method
2 //example 7.8
3 //page 273
4 clc;clear;close;

```



```

5 A=[2,3,1;1,2,3;3,1,2];
6 B=[9;6;8]
7 L(1,2)=0,L(1,3)=0,L(2,3)=0;
8 U(2,1)=0,U(3,1)=0,U(3,2)=0;
9 for i=1:3
10     L(i,i)=1;
11 end
12 for i=1:3
13     U(1,i)=A(1,i);
14 end
15 L(2,1)=1/U(1,1);
16 for i=2:3
17     U(2,i)=A(2,i)-U(1,i)*L(2,1);
18 end
19 L(3,1)=A(3,1)/U(1,1);
20 L(3,2)=(A(3,2)-U(1,2)*L(3,1))/U(2,2);
21 U(3,3)=A(3,3)-U(1,3)*L(3,1)-U(2,3)*L(3,2);
22 printf('The Matrix A in Triangle form\n \n')
23 printf('Matrix L\n');
24 for i=1:3
25     for j=1:3
26         printf('%.2f ',L(i,j));
27     end
28     printf('\n');
29 end
30 printf('\n \n');
31 printf('Matrix U\n');
32 for i=1:3
33     for j=1:3
34         printf('%.2f ',U(i,j));
35     end
36     printf('\n');
37 end
38 Y=L^-1*B;
39 X=U^-1*Y;
40 printf(' the values of x=%f,y=%f,z=%f',X(1,1),X(2,1)
    ,X(3,1));

```

---

### Scilab code Exa 7.9 ill conditioned linear systems

```
1 //ill conditioned linear systems
2 //example 7.9
3 //page 276
4 clc;clear;close;
5 A=[2 1;2 1.01];
6 B=[2;2.01];
7 X=A^-1*B;
8 A_e=0;
9 for i=1:2
10     for j=1:2
11         A_e=A_e+A(i,j)^2;
12     end
13 end
14 A_e=sqrt(A_e);
15 inv_A=A^-1;
16 invA_e=0;
17 for i=1:2
18     for j=1:2
19         invA_e=invA_e+inv_A(i,j)^2;
20     end
21 end
22 invA_e=sqrt(invA_e);
23 C=A_e*invA_e
24 de_A=determ(A);
25 for i=1:2
26     s=0;
27     for j=1:2
28         s=s+A(i,j)^2
29     end
30     s=sqrt(s);
31     k=de_A/s;
32 end
```

```

33 if k<1 then
34     printf(' the fuction is ill conditioned')
35 end

```

---

**Scilab code Exa 7.10** ill condiioned linear systems

```

1 //ill condiioned linear systems
2 //example 7.10
3 //page 277
4 clc;clear;close;
5 A=[1/2 1/3 1/4;1/5 1/6 1/7;1/8 1/9 1/10]//hilbert 's
   matrix
6 de_A=det(A);
7 if de_A<1 then
8     printf(' A is ill-conditioned')
9 end

```

---

**Scilab code Exa 7.11** ill conditioned linear systems

```

1 //ill conditioned linear system
2 //example 7.11
3 //page 277
4 clc;clear;close;
5 A=[25 24 10;66 78 37;92 -73 -80];
6 de_A=det(A);
7 for i=1:3
8     s=0;
9     for j=1:3
10        s=s+A(i,j)^2
11    end
12    s=sqrt(s);
13    k=de_A/s;
14 end

```

```

15 if k<1 then
16     printf(' the fuction is ill conditioned')
17 end

```

---

**Scilab code Exa 7.12** ill conditioned system

```

1 //ill-conditioned system
2 //example 7.12
3 //page 278
4 clc;clear;close;
5 //the original equations are 2x+y=2    2x+1.01y=2.01
6 A1=[2 1;2 1.01];
7 C1=[2;2.01];
8 x1=1;y1=1//approximate values
9 A2=[2 1;2 1.01];
10 C2=[3;3.01];
11 C=C1-C2;
12 X=A1^-1*C;
13 x=X(1,1)+x1;
14 y=X(2,1)+y1;
15 printf(' the exact solution is X=%f \t Y=%f',x,y);

```

---

**Scilab code Exa 7.14** solution of equations by iteration method

```

1 //solution of equations by iteration method
2 //example 7.14
3 //page 282
4 //jacobi's method
5 clc;clear;close;
6 C=[3.333;1.5;1.4];
7 X=[3.333;1.5;1.4];
8 B=[0 -0.1667 -0.1667;-0.25 0 0.25;-0.2 0.2 0];
9 for i=1:10

```

```

10     X1=C+B*X;
11     printf('X%d',i);
12     for k=1:3
13         for l=1:1
14             printf(' %f ',X1(k,l))
15         end
16         printf('\n');
17     end
18     X=X1;
19 end
20 printf(' the solution of the equation is converging
    at 3 1 1\n\n');
21 //gauss-seidel method
22 C=[3.333;1.5;1.4];
23 X=[3.333;1.5;1.4];
24 B=[0 -0.1667 -0.1667;-0.25 0 0.25;-0.2 0.2 0];
25 X1=C+B*X;
26 x=X1(1,1);y=X1(2,1);z=X1(3,1);
27 for i=1:5
28     x=3.333-0.1667*y-0.1667*z
29     y=1.5-0.25*x+0.25*z
30     z=1.4-0.2*x+0.2*y
31     printf(' the value after %d iteration is : %f\t
    %f\t %f\t\n\n',i,x,y,z)
32 end
33 printf(' again we conclude that roots converges at 3
    1 1')

```

---

### Scilab code Exa 7.15 eigenvalues and eigenvectors

```

1 //eigenvalues and eigenvectors
2 //example 7.15
3 //page 285
4 clc;clear;close
5 A=[5 0 1;0 -2 0;1 0 5];

```

```

6 x=poly(0, 'x');
7 for i=1:3
8     A(i,i)=A(i,i)-x;
9 end
10 d=determ(A);
11 X=roots(d);
12 printf(' the eigen values are \n\n')
13 disp(X);
14 X1=[0;1;0]
15 X2=[1/sqrt(2);0;-1/sqrt(2)];
16 X3=[1/sqrt(2);0;1/sqrt(2)];
17 //after computation the eigen vectors
18 printf('the eigen vectors for value %0.2g is ',X(3));
19 disp(X1);
20 printf('the eigen vectors for value %0.2g is ',X(2));
21 disp(X2);
22 printf('the eigen vectors for value %0.2g is ',X(1));
23 disp(X3);

```

---

#### Scilab code Exa 7.16 largest eigenvalue and eigenvectors

```

1 //largest eigenvalue and eigenvectors
2 //example 7.16
3 //page 286
4 clc;clear;close;
5 A=[1 6 1;1 2 0;0 0 3];
6 I=[1;0;0]; //initial eigen vector
7 X0=A*I
8 disp(X0, 'X0=')
9 X1=A*X0;
10 disp(X1, 'X1=')
11 X2=A*X1;
12 disp(X2, 'X2=')
13 X3=X2/3;
14 disp(X3, 'X3=')

```

```

15 X4=A*X3;
16 X5=X4/4;
17 disp(X5,'X5=');
18 X6=A*X5;
19 X7=X6/(4*4);
20 disp(X7,'X7=');
21 printf('as it can be seen that highest eigen value
        is 4 \n\n the eigen vector is %d %d %d',X7(1),X7
        (2),X7(3));

```

---

#### Scilab code Exa 7.17 housrholders method

```

1 //housrholder 's method
2 //example 7.17
3 //page 290
4 clc;clear;close;
5 A=[1 3 4;3 2 -1;4 -1 1];
6 S=sqrt(A(1,2)^2+A(1,3)^2);
7 v2=sqrt([1+(A(1,2)/S)]/2)
8 v3=A(1,3)/(2*v2*S)
9 V=[0 v2 v3];
10 P1=[1 0 0;0 1-2*v2^2 -2*v2*v3;0 -2*v2*v3 1-2*v3^2];
11 A1=P1*A*P1;
12 printf(' the reduced matrix is \n\n');
13 for i=1:3
14     for j=1:3
15         printf('%0.2f ',A1(i,j));
16     end
17     printf('\n');
18 end

```

---

#### Scilab code Exa 7.18 single value decommposition

```

1 //single value decommmposition
2 //example 7.18
3 //page 292
4 clc;clear;close;
5 A=[1 2;1 1;1 3];
6 A1=A'*A;
7 x=poly(0,'x');
8 A1(1,1)=A1(1,1)-x;
9 A1(2,2)=A1(2,2)-x;
10 de_A1=det(A1);
11 C=roots(de_A1);
12 printf(' eigen values are %0.2f %0.2f\n\n',C(1),C
    (2));
13 X1=[0.4033;0.9166];
14 X2=[0.9166;-0.4033];
15 Y1=(A*X1)/sqrt(C(1));
16 Y2=(A*X2)/sqrt(C(2));
17 printf(' singular decomposition of A is given by \n\
    n');
18 D1=[Y1(1) Y2(1);Y1(2) Y2(2);Y1(3) Y2(3)];
19 D2=[sqrt(C(1)) 0;0,sqrt(C(2))];
20 D3=[X1(1) X2(1);X1(2) X2(2)];
21 for i=1:3
22     for j=1:2
23         printf('%0.4f ',D1(i,j))
24     end
25     printf('\n')
26 end
27 printf('\n\n')
28 for i=1:2
29     for j=1:2
30         printf('%0.4f ',D2(i,j))
31     end
32     printf('\n')
33 end
34 printf('\n\n');
35 for i=1:2
36     for j=1:2

```



```
37         printf( '%0.4f      ', D3(i,j))
38     end
39     printf( '\n' )
40 end
```

---

# Chapter 8

## Numerical Solution of ordinary differential equation

Scilab code Exa 8.1 taylors method

```
1 //example 8.1
2 //taylor 's method
3 //page 304
4 clc;clear;close;
5 f=1;//value of function at 0
6 deff('z=f1(x)', 'z=x-f^2');
7 deff('z=f2(x)', 'z=1-2*f*f1(x)');
8 deff('z=f3(x)', 'z=-2*f*f2(x)-2*f2(x)^2');
9 deff('z=f4(x)', 'z=-2*f*f3(x)-6*f1(x)*f2(x)');
10 deff('z=f5(x)', 'z=-2*f*f4(x)-8*f1(x)*f3(x)-6*f2(x)^2
    ');
11 h=0.1;//value at 0.1
12 k=f;
13     for j=1:5
14         if j==1 then
15             k=k+h*f1(0);
16         elseif j==2 then
17             k=k+(h^j)*f2(0)/factorial(j)
18         elseif j ==3
```

```

19         k=k+(h^j)*f3(0)/factorial(j)
20     elseif j ==4
21         k=k+(h^j)*f4(0)/factorial(j)
22     elseif j==5
23         k=k+(h^j)*f5(0)/factorial(j)
24
25 end
26 end
27 printf('the value of the function at %.2f is :%.4f'
        ,h,k)

```

---

#### Scilab code Exa 8.2 taylors method

```

1 //taylor 's method
2 //example 8.2
3 //page 304
4 clc;clear;close;
5 f=1;//value of function at 0
6 f1=0;//value of first derivatie at 0
7 deff('y=f2(x)', 'y=x*f1+f');
8 deff('y=f3(x)', 'y=x*f2(x)+2*f1');
9 deff('y=f4(x)', 'y=x*f3(x)+3*f2(x)');
10 deff('y=f5(x)', 'y=x*f4(x)+4*f3(x)');
11 deff('y=f6(x)', 'y=x*f5(x)+5*f4(x)');
12 h=0.1;//value at 0.1
13 k=f;
14     for j=1:6
15         if j==1 then
16             k=k+h*f1;
17         elseif j==2 then
18             k=k+(h^j)*f2(0)/factorial(j)
19         elseif j ==3
20             k=k+(h^j)*f3(0)/factorial(j)
21         elseif j ==4
22             k=k+(h^j)*f4(0)/factorial(j)

```

```

23             elseif j==5
24                 k=k+(h^j)*f5(0)/factorial(j)
25             else
26 k=k+(h^j)*f6(0)/factorial (j)
27 end
28 end
29 printf('the value of the function at %.2f is :%0.7f'
        ,h,k)

```

---

### Scilab code Exa 8.3 picards method

```

1 //example 8.3
2 //picard 's method
3 //page 306
4 clc;clear;close;
5 deff('z=f(x,y)', 'z=x+y^2')
6 y(1)=1;
7 for i=1:2
8     y(i+1)=y(1)+integrate('f(x,y(i))','x',0,i /10);
9     printf ( ' \n y (%g) = %g\n ' ,i/10 ,y(i +1) );
10    end

```

---

### Scilab code Exa 8.4 picards method

```

1 //example 8.4
2 //picard 's method
3 //page 306
4 clc;clear;close;
5 deff('z=f(x,y)', 'z=x^2/(y^2+1)')
6 y(1)=0; //value at 0
7 c=0.25;
8 for i=1:3
9     y(i+1)=y(1)+integrate('f(x,y(i))','x',0,c);

```

```
10     printf ( ' \n y(%0.2f) = %g\n',c ,y(i +1) );
11     c=c*2;
12     end
```

---

### Scilab code Exa 8.5 eulers method

```
1 //example 8.5
2 //euler 's method
3 //page 308
4 clc;clear;close;
5 deff('z=f(y)', 'z=-y')
6 y(1)=1; //value at 0
7 h=0.01;c=0.01;
8 for i=1:4
9     y(i+1)=y(i)+h*f(y(i))
10    printf ( '\ny(%g)=%g\n',c,y(i+1));
11    c=c+0.01;
12 end
```

---

### Scilab code Exa 8.6 error estimates in eulers

```
1 //example 8.6
2 //error estimates in euler 's
3 //page 308
4 clc;clear;close;
5 deff('z=f(y)', 'z=-y')
6 y(1)=1; //value at 0
7 h=0.01;c=0.01;
8 for i=1:4
9     y(i+1)=y(i)+h*f(y(i))
10    printf ( '\ny(%g)=%g\n',c,y(i+1));
11    c=c+0.01;
12 end
```

```

13 for i=1:4
14     L(i)=abs(-(1/2)*h^2*y(i+1));
15     printf('L(%d) =%f\n\n',i,L(i))
16 end
17 e(1)=0;
18 for i=1:4
19     e(i+1)=abs(y(2)*e(i)+L(1));
20     printf('e(%d)=%f\n\n',i,e(i))
21 end
22 Actual_value=exp(-0.04);
23 Estimated_value=y(5);
24 err=abs(Actual_value-Estimated_value);
25 if err<e(5) then
26     disp(' VERIFIED ');
27 end

```

---

### Scilab code Exa 8.7 modified eulers method

```

1 //example 8.7
2 //modified euler 's method
3 //page 310
4 clc;clear;close;
5 h=0.05;
6 f=1;
7 deff('z=f1(x,y)', 'z=x^2+y');
8 x=0:0.05:0.1
9 y1=0;
10 y1(1)=f+h*f1(x(1),f);
11 y1(2)=f+h*(f1(x(1),f)+f1(x(2),y1(1)))/2;
12 y1(3)=f+h*(f1(x(1),f)+f1(x(3),y1(2)))/2;
13 y2(1)=y1(2)+h*f1(x(2),y1(2));
14 y2(2)=y1(2)+h*(f1(x(2),y1(2))+f1(x(3),y2(1)))/2;
15 y2(3)=y1(2)+h*(f1(x(2),y1(2))+f1(x(3),y2(2)))/2;
16 printf(' y1(0)\t y1(1)\t y1(2)\t y2(0)\t y2(1)\t y3
(2)\n\n' );

```

```

17 printf( ' %f\t      %f\t      %f\t      %f\t      %f\t      %f
    \n', y1(1), y1(2), y1(3), y2(1), y2(2), y2(3) )
18 printf( '\n\n the value of y at %0.2f is : %0.4f', x
    (3), y2(3) );

```

---

### Scilab code Exa 8.8 runge kutta formula

```

1 //example 8.8
2 //runge-kutta formula
3 //page 313
4 clc;clear;close;
5 deff( 'y=f(x,y)', 'y=y-x' )
6 y=2;x=0;h=0.1;
7 K1=h*f(x,y);
8 K2=h*f(x+h,y+K1);
9 y1=y+( K1+K2)/2
10 printf( '\n y(0.1) by second order runge kutta
    method:%0.4f', y1);
11 y=y1;x=0.1;h=0.1;
12 K1=h*f(x,y);
13 K2=h*f(x+h,y+K1);
14 y1=y+( K1+K2)/2
15 printf( '\n y(0.2) by second order runge kutta
    method:%0.4f', y1);
16 y=2,x=0,h=0.1;
17 K1=h*f(x,y);
18 K2=h*f(x+h/2,y+K1/2);
19 K3=h*f(x+h/2,y+K2/2);
20 K4=h*f(x+h,y+K3);
21 y1=y+(K1+2*K2+2*K3+K4)/6;
22 printf( '\n y(0.1) by fourth order runge kutta
    method:%0.4f', y1);
23 y=y1,x=0.1,h=0.1;
24 K1=h*f(x,y);
25 K2=h*f(x+h/2,y+K1/2);

```

```

26 K3=h*f(x+h/2,y+K2/2);
27 K4=h*f(x+h,y+K3);
28 y1=y+(K1+2*K2+2*K3+K4)/6;
29 printf ('\n y(0.1) by fourth order runge kutta
        method:%0.4 f ',y1);y=2,x=0,h=0.1;

```

---

### Scilab code Exa 8.9 runge kutta formula

```

1 //example 8.9
2 //runge kutta method
3 //page 315
4 clc;clear;close;
5 deff('y=f(x,y)', 'y=1+y^2');
6 y=0,x=0,h=0.2;
7 K1=h*f(x,y);
8 K2=h*f(x+h/2,y+K1/2);
9 K3=h*f(x+h/2,y+K2/2);
10 K4=h*f(x+h,y+K3);
11 y1=y+(K1+2*K2+2*K3+K4)/6;
12 printf ('\n y(0.2) by fourth order runge kutta
        method:%0.4 f ',y1);
13 y=y1,x=0.2,h=0.2;
14 K1=h*f(x,y);
15 K2=h*f(x+h/2,y+K1/2);
16 K3=h*f(x+h/2,y+K2/2);
17 K4=h*f(x+h,y+K3);
18 y1=y+(K1+2*K2+2*K3+K4)/6;
19 printf ('\n y(0.4) by fourth order runge kutta
        method:%0.4 f ',y1);
20 y=2,x=0,h=0.1;
21 y=y1,x=0.4,h=0.2;
22 K1=h*f(x,y);
23 K2=h*f(x+h/2,y+K1/2);
24 K3=h*f(x+h/2,y+K2/2);
25 K4=h*f(x+h,y+K3);

```



```

26 y1=y+(K1+2*K2+2*K3+K4)/6;
27 printf ( '\n y(0.6) by fourth order runge kutta
    method:%0.4 f ',y1);

```

---

### Scilab code Exa 8.10 initial value problems

```

1 //example 8.10
2 //initial value problems
3 //page 315
4 clc;clear;close;
5 deff('y=f1(x,y)', 'y=3*x+y/2');
6 y(1)=1;
7 h=0.1;c=0;
8 for i=1:2
9     y(i+1)=y(i)+h*f1(c,y(i))
10    printf ( '\ny(%g)=%g\n',c,y(i))
11    c=c+0.1;
12 end

```

---

### Scilab code Exa 8.11 adams moulton method

```

1 //example 8.11
2 //adam's moulton method
3 //page 316
4 clc;clear;close;
5 deff('y=f(x,y)', 'y=1+y^2');
6 y=0,x=0,h=0.2,f1(1)=0;
7 K1=h*f(x,y);
8 K2=h*f(x+h/2,y+K1/2);
9 K3=h*f(x+h/2,y+K2/2);
10 K4=h*f(x+h,y+K3);
11 y1=y+(K1+2*K2+2*K3+K4)/6;
12 f1(1)=y1;

```

```

13 printf ( '\n y(0.2) by fourth order runge kutta
    method:%0.4 f ',y1);
14 y=y1,x=0.2,h=0.2;
15 K1=h*f(x,y);
16 K2=h*f(x+h/2,y+K1/2);
17 K3=h*f(x+h/2,y+K2/2);
18 K4=h*f(x+h,y+K3);
19 y1=y+(K1+2*K2+2*K3+K4)/6;
20 f1(2)=y1;
21 printf ( '\n y(0.4) by fourth order runge kutta
    method:%0.4 f ',y1);
22 y=2,x=0,h=0.1;
23 y=y1,x=0.4,h=0.2;
24 K1=h*f(x,y);
25 K2=h*f(x+h/2,y+K1/2);
26 K3=h*f(x+h/2,y+K2/2);
27 K4=h*f(x+h,y+K3);
28 y1=y+(K1+2*K2+2*K3+K4)/6;
29 f1(3)=y1;
30 printf ( '\n y(0.6) by fourth order runge kutta
    method:%0.4 f ',y1);
31 y_p=y1+h*(55*(1+f1(3)^2)-59*(1+f1(2)^2)+37*(1+f1(1)
    ^2)-9)/24;
32 y_c=y1+h*(9*(1+(y_p-1)^2)+19*(1+f1(3)^2)-5*(1+f1(2)
    ^2)+(1+f1(1)^2))/24;
33 printf ( ' \nthe predicted value is:%0.4 f:\n ',y_p);
34 printf ( ' the computed value is:%0.4 f:',y_c);

```

---

### Scilab code Exa 8.12 milnes method

```

1 //example 8.12
2 //milne's method
3 //page 320
4 clc;clear;close;
5 deff('y=f(x,y)', 'y=1+y^2');

```

```

6 y=0,x=0,h=0.2,f1(1)=0;
7 printf('x          y          y1=1+y^2\n\n
      ')
8 Y1(1)=1+y^2;
9 printf('%0.4f          %0.4f          %0.4f\n',x,y,(1+y
      ^2));
10 K1=h*f(x,y);
11 K2=h*f(x+h/2,y+K1/2);
12 K3=h*f(x+h/2,y+K2/2);
13 K4=h*f(x+h,y+K3);
14 y1=y+(K1+2*K2+2*K3+K4)/6;
15 f1(1)=y1;
16 Y1(2)=1+y1^2;
17 printf('%0.4f          %0.4f          %0.4f\n',x+h,y1,(1+
      y1^2));
18 y=y1,x=0.2,h=0.2;
19 K1=h*f(x,y);
20 K2=h*f(x+h/2,y+K1/2);
21 K3=h*f(x+h/2,y+K2/2);
22 K4=h*f(x+h,y+K3);
23 y1=y+(K1+2*K2+2*K3+K4)/6;
24 f1(2)=y1;
25 Y1(3)=1+y1^2
26 printf('%0.4f          %0.4f          %0.4f\n',x+h,y1,(1+
      y1^2));
27 y=y1,x=0.4,h=0.2;
28 K1=h*f(x,y);
29 K2=h*f(x+h/2,y+K1/2);
30 K3=h*f(x+h/2,y+K2/2);
31 K4=h*f(x+h,y+K3);
32 y1=y+(K1+2*K2+2*K3+K4)/6;
33 f1(3)=y1;
34 Y1(4)=1+y1^2;
35 printf('%0.4f          %0.4f          %0.4f\n',x+h,y1,(1+
      y1^2));
36 Y_4=4*h*(2*Y1(2)-Y1(3)+2*Y1(4))/3;
37 printf('y(0.8)=%f\n',Y_4);
38 Y=1+Y_4^2;

```

```

39 Y_4=f1(2)+h*(Y1(3)+4*Y1(4)+Y)/3; //more correct value
40 printf('y(0.8)=%f\n',Y_4);

```

---

### Scilab code Exa 8.13 milnes method

```

1 //example 8.13
2 //milne's method
3 //page 320
4 clc;clear;close;
5 deff('y=f1(x,y)', 'y=x^2+y^2-2');
6 x=[-0.1 0 0.1 0.2];
7 y=[1.0900 1.0 0.8900 0.7605];
8 h=0.1;
9 for i=1:4
10     Y1(i)=f1(x(i),y(i));
11 end
12 printf('      x              y          y1=x^2+
      y^2-2      \n\n');
13 for i=1:4
14     printf('    %0.2f          %f          %f
      \n',x(i),y(i),Y1(i));
15 end
16 Y_3=y(1)+(4*h/3)*(2*Y1(2)-Y1(3)+2*Y1(4));
17 printf('y(0.3)=%f\n',Y_3)
18 Y1_3=f1(0.3,Y_3);
19 Y_3=y(3)+h*(Y1(3)+4*Y1(4)+Y1_3)/3; //corrected value
20 printf('corrected y(0.3)=%f',Y_3)

```

---

### Scilab code Exa 8.14 initial value problems

```

1 //example 8.14
2 //initial-value problem
3 //page 322

```

```

4 clc;clear;close;
5 deff('y=f(x)', 'y=13*exp(x/2)-6*x-12');
6 s1=1.691358;s3=3.430879;
7 printf('the error in the computed values are %0.7g
    %0.7g', abs(f(0.5)-s1), abs(f(1)-s3))

```

---

**Scilab code Exa 8.15** boundary value problem using finite difference method

```

1 //boundary value problem using finite difference
   method
2 //example 8.15
3 //page 328
4 clc;clear;close;
5 deff('y=f(x)', 'y=cos(x)+((1-cos(1))/sin(1))*sin(x)-1
    ');
6 h1=1/2;
7 Y=f(0.5);
8 y0=0,y2=0;
9 y1=4*(1/4+y0+y2)/7
10 printf('computed value with h=%f of y(0.5) is %f\n',
    h1,y1)
11 printf('error in the result with actual value %f\n',
    abs(Y-y1) )
12 h2=1/4;
13 y0=0,y4=0;
14 //solving the approximated differential equation
15 A=[-31/16 1 0;1 -31/16 1;0 1 -31/16];
16 X=[-1/16;-1/16;-1/16];
17 C=A^-1*X;
18 printf('computed value with h=%f of y(0.5) is %f\n',
    h2,C(2,1))
19 printf('error in the result with actual value %f\n',
    abs(Y-C(2,1)))

```

---

**Scilab code Exa 8.16** boundary value problem using finite difference method

```
1 //boundary value problem using finite difference
  method
2 //example 8.16
3 //page 329
4 clc; clear; close;
5 deff( 'y=f(x) ', 'y=sinh(x) ')
6 y0=0//y(0)=0;
7 y4=3.62686//y(2)=3.62686
8 h1=0.5;
9 Y=f(0.5)
10 //arranging and calculating the values
11 A=[-9 4 0;4 -9 4;0 4 -9];
12 C=[0;0;-14.50744];
13 X=A^-1*C
14 printf('computed value with h=%f of y(0.5) is %f\n',
  h1,X(1,1))
15 printf('error in the result with actual value %f\n',
  abs(Y-X(1,1)) )
16 h2=1.0;
17 y0=0//y(0)=0;
18 y2=3.62686//y(2)=3.62686
19 y1=(y0+y2)/3;
20 Y=(4*X(2,1)-y1)/3;
21 printf('with better approximation error is reduced
  to %f',abs(Y-f(1.0)));
```

---

**Scilab code Exa 8.17** cubic spline method

```
1 //cubic spline method
2 //example 8.17
```

```

3 //page 330
4 clc;clear;close;
5 deff('y=f(x)', 'y=cos(x)+((1-cos(1))/sin(1))*sin(x)-1
      ');
6 y=[f(0) f(0.5) f(1)];
7 h=1/2;
8 Y=f(0.5);
9 y0=0,y2=0;
10 M0=-1;M1=-22/25;M2=-1;
11 s1_0=47/88;s1_1=-47/88;s1_05=0;
12 printf('the cubic spline value is %f',Y)

```

---

**Scilab code Exa 8.18** cubic spline method

```

1 //cubic spline method
2 //example 8.18
3 //page 331
4 clc;clear;close;
5 //after arranging and forming equation
6 A=[10 -1 0 24;0 16 -1 -32;1 20 0 16;0 1 26 -24];
7 C=[36;-12;24;-9];
8 X=A^-1*C;
9 printf(' Y1=%f\n\n',X(4,1));
10 printf(' the error in the solution is :%f',abs((2/3)
      -X(4,1)))

```

---

**Scilab code Exa 8.19** boundary value problem by cubic spline method

```

1 //boundary value problem by cubisc spline method
2 //example 8.18
3 //page 331
4 clc;clear;close;
5 h=1/2;

```

```
6 //arranging in two subintervals we get
7 A=[10 -1 0 24;0 16 -1 -32;1 20 0 16;0 1 26 -24];
8 C=[36;-12;24;-9];
9 X=A^-1*C
10 printf('the computed value of y(1.5) is %f ',X(4,1))
    ;
```

---



## Chapter 9

# Numerical Solution of Partial Differential Equation

Scilab code Exa 9.1 standard five point formula

```
1 //standard five point formula
2 //example 9.1
3 //page 350
4 clc;clear;close;
5 u2=5;u3=1;
6 for i=1:3
7     u1=(u2+u3+6)/4;
8     u2=u1/2+5/2;
9     u3=u1/2+1/2;
10    printf(' the values are u1=%d\t u2=%d\t u3=%d\t \n\n',u1,u2,u3);
11 end
```

---

Scilab code Exa 9.2 solution of laplace equation by jacobi method gauss seidel method and SOR method

```

1 //solution of laplace equation by jacobi method,
   gauss-seidel method and SOR method
2 //example 9.2
3 //page 351
4 clc;clear;close;
5 u1=0.25;u2=0.25;u3=0.5;u4=0.5;//initial values
6 printf('jacobis iteration process\n\n')
7 printf('u1\t      u2\t      u3\t      u4\t      \n\n')
8 printf('%f\t      %f\t      %f\t      %f\t      \n',u1,u2,
   u3,u4)
9 for i=1:7
10     u11=(0+u2+0+u4)/4
11     u22=(u1+0+0+u3)/4;
12     u33=(1+u2+0+u4)/4;
13     u44=(1+0+u3+u1)/4;
14     u1=u11;u2=u22;u3=u33;u4=u44;
15 printf('%f\t      %f\t      %f\t      %f\t      \n',u11,u22
   ,u33,u44)
16 end
17 printf(' gauss seidel process\n\n');
18 u1=0.25;u2=0.3125;u3=0.5625;u4=0.46875;//initial
   values
19 printf('u1\t      u2\t      u3\t      u4\t      \n\n')
20 printf('%f\t      %f\t      %f\t      %f\t      \n',u1,u2,
   u3,u4)
21 for i=1:4
22     u1=(0+u2+0+u4)/4
23     u2=(u1+0+0+u3)/4;
24     u3=(1+u2+0+u4)/4;
25     u4=(1+0+u3+u1)/4;
26     printf('%f\t      %f\t      %f\t      %f\t      \n',u1,
   u2,u3,u4)
27 end
28 printf('u1\t      u2\t      u3\t      u4\t      \n\n')
29 printf('%f\t      %f\t      %f\t      %f\t      \n',u1,u2,
   u3,u4)

```

---

### Scilab code Exa 9.4 poisson equation

```
1 //poisson equation
2 //example 9.4
3 //page 354
4 clc;clear;close;
5 u2=0;u4=0;
6 printf(' u1\t u2\t u3\t u4\t\n\n');
7 for i=1:6
8     u1=u2/2+30;
9     u2=(u1+u4+150)/4;
10    u4=u2/2+45;
11    printf(' %0.2f\t %0.2f\t %0.2f\t %0.2f\n',
12           u1,u2,u2,u4);
13 end
14 printf(' from last two iterates we conclude u1=67
15        u2=75 u3=75 u4=83\n')
```

---

### Scilab code Exa 9.6 bender schmidt formula

```
1 //bender-schmidt formula
2 //example 9.6
3 //page 362
4 clc;clear;close;
5 deff('y=f(x)', 'y=4*x-x^2');
6 u=[f(0) f(1) f(2) f(3) f(4)];
7 u11=(u(1)+u(3))/2;
8 u12=(u(2)+u(4))/2;
9 u13=(u(3)+u(5))/2;
10 printf(' u11=%0.2f\t u12=%0.2f\t u13=%0.2f\t \n',
11         u11,u12,u13)
11 u21=(u(1)+u12)/2;
```

```

12 u22=(u11+u13)/2;
13 u23=(u12+0)/2;
14 printf(' u21=%0.2 f\t u22=%0.2 f\t u23=%0.2 f\t \n',
    u21, u22, u23)
15 u31=(u(1)+u22)/2;
16 u32=(u21+u23)/2;
17 u33=(u22+u(1))/2;
18 printf(' u31=%0.2 f\t u32=%0.2 f\t u33=%0.2 f\t \n',
    u31, u32, u33)
19 u41=(u(1)+u32)/2;
20 u42=(u31+u33)/2;
21 u43=(u32+u(1))/2;
22 printf(' u41=%0.2 f\t u42=%0.2 f\t u43=%0.2 f\t \n',
    u41, u42, u43)
23 u51=(u(1)+u42)/2;
24 u52=(u41+u43)/2;
25 u53=(u42+u(1))/2;
26 printf(' u51=%0.2 f\t u52=%0.2 f\t u53=%0.2 f\t \n',
    u51, u52, u53)

```

---

**Scilab code Exa 9.7** bender schimdt's formula and crank nicolson formula

```

1 //bender-schimdt's formula and crank-nicolson
  formula
2 //example 9.7
3 //page 363
4 //bender -schimdt's formula
5 clc;clear;close;
6 deff('y=f(x,t)', 'y=exp(-%pi^2*t)*sin(%pi*x)');
7 u=[f(0,0) f(0.2,0) f(0.4,0) f(0.6,0) f(0.8,0) f(1,0)
  ];
8 u11=u(3)/2;u12=(u(2)+u(4))/2;u13=u12;u14=u11;
9 printf(' u11=%f\t u12=%f\t u13=%f\t u14=%f\n\n',
    u11, u12, u13, u14)
10 u21=u12/2;u22=(u12+u14)/2;u23=u22;u24=u21;

```

```

11 printf(' u21=%f\t u22=%f\t u23=%f\t u24=%f\n\n',
    u21, u22, u23, u24)
12 printf(' the error in the solution is: %f\n\n', abs(
    u22-f(0.6,0.04)))
13 //crank-nicolson formula
14 //by putting i=1,2,3,4 we obtain four equation
15 A=[4 -1 0 0 ; -1 4 -1 0; 0 -1 4 -1; 0 0 -1 4];
16 C=[0.9510; 1.5388; 1.5388; 0.9510];
17 X=A^-1*C;
18 printf(' u11=%f\t u21=%f\t u31=%f\t u41=%f\t\n\n',
    X(1,1), X(2,1), X(3,1), X(4,1))
19 printf(' the error in the solution is: %f\n\n', abs(X
    (2,1)-f(0.6,0.04)))

```

---

**Scilab code Exa 9.8** heat equation using crank nicolson method

```

1 //heat equation using crank-nicolson method
2 //example 9.8
3 //page 364
4 clc; clear; close;
5 U=0.01878;
6 //h=1/2; l=1/8, i=1;
7 u01=0; u21=1/8;
8 u11=(u21+u01)/6;
9 printf(' u11=%f\n\n', u11);
10 printf(' error is %f\n\n', abs(u11-U));
11 //h=1/4, l=1/8, i=1,2,3
12 A=[-3 -1 0; 1 -3 1; 0 1 -3];
13 C=[0; 0; -1/8];
14 X=A^-1*C;
15 printf(' u12=%f\n\n', X(2,1));
16 printf(' error is %f\n\n', abs(X(2,1)-U));

```

---