Scilab Textbook Companion for Numerical Methods For Engineers by S. C. Chapra And R. P. Canale[1]

Created by
Meghana Sundaresan
Numerical Methods
Chemical Engineering
VNIT
College Teacher
Kailas L. Wasewar
Cross-Checked by

July 31, 2019

[1]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: Numerical Methods For Engineers

Author: S. C. Chapra And R. P. Canale

Publisher: McGraw Hill, New York

Edition: 5

Year: 2006

ISBN: 0071244298

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

List of Figures

Chapter 1

Mathematical Modelling and Engineering Problem Solving

Scilab code Exa 1.1 Analytical Solution to Falling Parachutist Problem

```
clc;
clear;
g=9.8;//m/s^2; acceleration due to gravity
m=68.1;//kg
c=12.5;//kg/sec; drag coefficient
count=1;
for i=0:2:12
    v(count)=g*m*(1-exp(-c*i/m))/c;
    disp(v(count),"v(m/s)=",i,"Time(s)=")
    count=count+1;
end
disp(g*m/c,"v(m/s)=","infinity","Time(s)=")
```

Scilab code Exa 1.2 Numerical Solution to Falling Parachutist Problem

```
clc;
```

```
clear;
g=9.8;//m/s^2; acceleration due to gravity
m=68.1;//kg
c=12.5;//kg/sec; drag coefficient
count=2;
v(1)=0;
disp(v(1),"v(m/s)=",0,"Time(s)=")
for i=2:2:12
    v(count)=v(count-1)+(g-c*v(count-1)/m)*(2);
    disp(v(count),"v(m/s)=",i,"Time(s)=")
    count=count+1;
end
disp(g*m/c,"v(m/s)=","infinity","Time(s)=")
```

Chapter 2

Programming and Software

Scilab code Exa 2.1 roots of quadratic

```
clc;
clear;
response=1;
while response==1
a=input("Input the value of a:")
b=input("Input the value of b:")
c=input("Input the value of c:")
if a==0 then
    if b~=0 then
        r1=-c/b;
        disp(r1,"The root:")
    else disp("Trivial Solution.")
    end
else
    discr=b^2-4*a*c;
    if discr>=0 then
        r1=(-b+sqrt(discr))/(2*a);
        r2=(-b-sqrt(discr))/(2*a);
        disp(r2,"and",r1,"The roots are:")
    else
        r1=-b/(2*a);
```

```
        r2=r1;
        i1=sqrt(abs(discr))/(2*a);
        i2=-i1;
        disp(r2+i2*sqrt(-1),r1+i1*sqrt(-1),"The
           roots are:")
    end
end
response=input("Do you want to continue (press 1 for
    yes and 2 for no)?")
if response=2 then exit;
end
end
end
```

Chapter 3

Approximations and Round off Errors

Scilab code Exa 3.1 Calculations of Errors

```
clc;
clear;
lbm=9999; //cm, measured length of bridge
lrm=9;//cm, measured length of rivet
lbt=10000;//cm, true length of bridge
lrt=10;//cm,true length of rivet
//calculating true error below;
Etb=lbt-lbm;//cm, true error in bridge
Etr=lrt-lrm;//cm, true error in rivet
//calculating percent relative error below
etb=Etb*100/lbt;//percent relative error for bridge
etr=Etb*100/lrt;//percent relative error for rivet
disp("a. The true error is")
disp("for the bridge","cm",Etb)
disp("for the rivet","cm",Etr)
disp("b. The percent relative error is")
disp("for the bridge","%",etb)
disp("for the rivet","%",etr)
```

Scilab code Exa 3.2 Iterative error estimation

```
clc;
clear;
n=3;//number of significant figures
es=0.5*(10^(2-n));//percent, specified error criterion
x=0.5;
f(1)=1;//first estimate f=e^x = 1
ft=1.648721;//true value of e^0.5=f
et(1)=(ft-f(1))*100/ft;
ea(1)=100;
i=2;
while ea(i-1)>=es
    f(i)=f(i-1)+(x^(i-1))/(factorial(i-1));
    et(i)=(ft-f(i))*100/ft;
    ea(i)=(f(i)-f(i-1))*100/f(i);
    i=i+1;
end
for j=1:i-1
   disp(ea(j),"Approximate estimate of error(%)=",et(j),"True % relative error=",f(j),"Result=",j,"term number=")
   disp("--------------------------------------------------------------------------------------------")
end
```

Scilab code Exa 3.3 Range of Integers

```
n=16;//no of bits
num=0;
```

```
for i=0:(n-2)
    num=num+(1*(2^i));
end
disp((-1*num),"to",num,"Thus a 16-bit computer word
   can store decimal integers ranging from")
```

Scilab code Exa 3.4 Floating Point Numbers

```
clc;
clear;
n=7;//no. of bits
//the maximum value of exponents is given by
Max=1*(2^1)+1*(2^0);
//mantissa is found by
mantissa=1*(2^-1)+0*(2^-3)+0*(2^-3);
num=mantissa*(2^(Max*-1));//smallest possible
   positive number for this system
disp(num,"The smallest possible positive number for
   this system is")
```

Scilab code Exa 3.5 Machine Epsilon

```
clc;
clear;
b=2;//base
t=3;//number of mantissa bits
E=2^(1-t);//epsilon
disp(E,"value of epsilon=")
```

Scilab code Exa 3.6 Interdependent Computations

```
clc;
clear;
num=input("Input a number: ")
sum=0;
for i=1:100000
    sum=sum+num;
end
disp(sum,"The number summed up 100,000 times is=")
```

Scilab code Exa 3.7 Subtractive Cancellation

```
clc;
clear;
a=1;
b=3000.001;
c=3;
//the roots of the quadratic equation x^2+3000.001*x
   +3=0 are found as
D=(b^2)-4*a*c;
x1=(-b+(D^0.5))/(2*a);
x2=(-b-(D^0.5))/(2*a);
disp(x2,and,x1,"The roots of the quadratic equation
   (x^2)+(3000.001*x)+3=0 are = ")
```

Scilab code Exa 3.8 Infinite Series Evaluation

```
clc;
clear;
function y=f(x)
    y=exp(x)
endfunction
sum=1;
test=0;
```

```
i=0;
term=1;
x=input("Input value of x:")
while sum~=test
    disp(sum,"sum:",term,"term:",i,"i:")
    disp("------------------------------------------")
    i=i+1;
    term=term*x/i;
    test=sum;
    sum=sum+term;
end
disp(f(x),"Exact Value")
```

Chapter 4

Truncation Errors and the Taylor Series

Scilab code Exa 4.1 `Polynomial Taylor Series`

```
clc;
clear;
function y=f(x)
    y=-0.1*x^4-0.15*x^3-0.5*x^2-0.25*x+1.2;
endfunction
xi=0;
xf=1;
h=xf-xi;
fi=f(xi);//function value at xi
ffa=f(xf);//actual function value at xf

//for n=0, i.e, zero order approximation
ff=fi;
Et(1)=ffa-ff;//truncation error at x=1
disp(fi,"The value of f at x=0 :")
disp(ff,"The value of f at x=1 due to zero order approximation :")
disp(Et(1),"Truncation error :")
disp("———————————————————————————————
```

```
")

//for n=1, i.e, first order approximation
deff('y=f1(x)','y=derivative(f,x,order=4)')
f1i=f1(xi);//value of first derivative of function at xi
f1f=fi+f1i*h;//value of first derivative of function at xf
Et(2)=ffa-f1f;//truncation error at x=1
disp(f1i,"The value of first derivative of f at x=0 :")
disp(f1f,"The value of f at x=1 due to first order approximation :")
disp(Et(2),"Truncation error :")
disp("----------------------------------------------------------------")


//for n=2, i.e, second order approximation
deff('y=f2(x)','y=derivative(f1,x,order=4)')
f2i=f2(xi);//value of second derivative of function at xi
f2f=f1f+f2i*(h^2)/factorial(2);//value of second derivative of function at xf
Et(3)=ffa-f2f;//truncation error at x=1
disp(f2i,"The value of second derivative of f at x=0 :")
disp(f2f,"The value of f at x=1 due to second order approximation :")
disp(Et(3),"Truncation error :")
disp("----------------------------------------------------------------")

//for n=3, i.e, third order approximation
deff('y=f3(x)','y=derivative(f2,x,order=4)')
f3i=f3(xi);//value of third derivative of function at xi
f3f=f2f+f3i*(h^3)/factorial(3);//value of third
```

```
derivative of function at xf
Et(4)=ffa-f3f;//truncation error at x=1
disp(f3i,"The value of third derivative of f at x=0 :")
disp(f3f,"The value of f at x=1 due to third order approximation :")
disp(Et(4),"Truncation error :")
disp("-------------------------------------------------")

//for n=4, i.e, fourth order approximation
deff('y=f4(x)','y=derivative(f3,x,order=4)')
f4i=f4(xi);//value of fourth derivative of function at xi
f4f=f3f+f4i*(h^4)/factorial(4);//value of fourth derivative of function at xf
Et(5)=ffa-f4f;//truncation error at x=1
disp(f4i,"The value of fourth derivative of f at x=0 :")
disp(f4f,"The value of f at x=1 due to fourth order approximation :")
disp(Et(5),"Truncation error :")
disp("-------------------------------------------------")
```

Scilab code Exa 4.2 Taylor Series Expansion

```
clc;
clear;
function y=f(x)
    y=cos(x)
endfunction
xi=%pi/4;
xf=%pi/3;
h=xf-xi;
```

```
fi=f(xi);//function value at xi
ffa=f(xf);//actual function value at xf

//for n=0, i.e, zero order approximation
ff=fi;
et(1)=(ffa-ff)*100/ffa;//percent relative error at x=1
disp(ff,"The value of f at x=1 due to zero order approximation :")
disp(et(1),"% relative error :")
disp("--------------------------------------------------")

//for n=1, i.e, first order approximation
deff('y=f1(x)','y=derivative(f,x,order=4)')
f1i=f1(xi);//value of first derivative of function at xi
f1f=fi+f1i*h;//value of first derivative of function at xf
et(2)=(ffa-f1f)*100/ffa;//% relative error at x=1
disp(f1f,"The value of f at x=1 due to first order approximation :")
disp(et(2),"% relative error :")
disp("--------------------------------------------------")


//for n=2, i.e, second order approximation
deff('y=f2(x)','y=derivative(f1,x,order=4)')
f2i=f2(xi);//value of second derivative of function at xi
f2f=f1f+f2i*(h^2)/factorial(2);//value of second derivative of function at xf
et(3)=(ffa-f2f)*100/ffa;//% relative error at x=1
disp(f2f,"The value of f at x=1 due to second order approximation :")
disp(et(3),"% relative error :")
disp("--------------------------------------------------
```

```
")


//for n=3, i.e, third order approximation
deff('y=f3(x)','y=derivative(f2,x,order=4)')
f3i=f3(xi);//value of third derivative of function at xi
f3f=f2f+f3i*(h^3)/factorial(3);//value of third derivative of function at xf
et(4)=(ffa-f3f)*100/ffa;//% relative error at x=1
disp(f3f,"The value of f at x=1 due to third order approximation :")
disp(et(4),"% relative error :")
disp("--------------------------------------------------------")


//for n=4, i.e, fourth order approximation
deff('y=f4(x)','y=derivative(f3,x,order=4)')
f4i=f4(xi);//value of fourth derivative of function at xi
f4f=f3f+f4i*(h^4)/factorial(4);//value of fourth derivative of function at xf
et(5)=(ffa-f4f)*100/ffa;//% relative error at x=1
disp(f4f,"The value of f at x=1 due to fourth order approximation :")
disp(et(5),"% relative error :")
disp("--------------------------------------------------------")


//for n=5, i.e, fifth order approximation
f5i=(f4(1.1*xi)-f4(0.9*xi))/(2*0.1);//value of fifth derivative of function at xi (central difference method)
f5f=f4f+f5i*(h^5)/factorial(5);//value of fifth derivative of function at xf
et(6)=(ffa-f5f)*100/ffa;//% relative error at x=1
```

```
disp(f5f,"The value of f at x=1 due to fifth order approximation :")
disp(et(6),"% relative error :")
disp("-----------------------------------------------------------------")


//for n=6, i.e, sixth order approximation
deff('y=f6(x)','y=derivative(f5,x,order=4)')
f6i=(f4(1.1*xi)-2*f4(xi)+f4(0.9*xi))/(0.1^2);//value of sixth derivative of function at xi (central difference method)
f6f=f5f+f6i*(h^6)/factorial(6);//value of sixth derivative of function at xf
et(6)=(ffa-f6f)*100/ffa;//% relative error at x=1
disp(f6f,"The value of f at x=1 due to sixth order approximation :")
disp(et(6),"% relative error :")
disp("-----------------------------------------------------------------")
```

Scilab code Exa 4.3 Effect of Nonlinearity and Stepsize on Taylor expansion

```
clc;
clear;
m=input("Input value of m:")
h=input("Input value of h:")
function y=f(x)
    y=x^m
endfunction
x1=1;
x2=x1+h;
fx1=f(x1);
fx2=fx1+m*(fx1^(m-1))*h;
if m==1 then
```

```
    R=0;
else if m==2 then
        R=2*(h^2)/factorial(2);
        end
      if m==3 then
        R=(6*(x1)*(h^2)/factorial(2))+(6*(h^3)/factorial(3));
        end
      if m==4 then
        R=(12*(x1^2)*(h^2)/factorial(2))+(24*(x1)*(h^3)/factorial(3))+(24*(h^4)/factorial(4));
        end
end
disp(R,"Remainder:",fx2,"The value by first order approximation:")
disp(f(x2),"True Value at x2:")
```

Scilab code Exa 4.4 Finite divided difference approximation of derivatives

```
clc;
clear;
function y=f(x)
    y=-0.1*(x^4)-0.15*(x^3)-0.5*(x^2)-0.25*(x)+1.2
endfunction
x=0.5;
h=input("Input h:")
x1=x-h;
x2=x+h;
//forward difference method
fdx1=(f(x2)-f(x))/h;//derivative at x
et1=abs((fdx1-derivative(f,x))/derivative(f,x))*100;
//backward difference method
fdx2=(f(x)-f(x1))/h;//derivative at x
et2=abs((fdx2-derivative(f,x))/derivative(f,x))*100;
```

```
//central difference method
fdx3=(f(x2)-f(x1))/(2*h);//derivative at x
et3=abs((fdx3-derivative(f,x))/derivative(f,x))*100;
disp(h,"For h=")
disp(et1,"and percent error=",fdx1,"Derivative at x by forward difference method=")
disp(et2,"and percent error=",fdx2,"Derivative at x by backward difference method=")
disp(et3,"and percent error=",fdx3,"Derivative at x by central difference method=")
```

Scilab code Exa 4.5 Error propagation in function of single variable

```
clc;
clear;
function y=f(x)
    y=x^3
endfunction
x=2.5;
delta=0.01;
deltafx=abs(derivative(f,x))*delta;
fx=f(x);
disp(fx+deltafx,"and",fx-deltafx,"true value is between")
```

Scilab code Exa 4.6 Error propagation in multivariable function

```
clc;
clear;
function y=f(F,L,E,I)
    y=(F*(L^4))/(8*E*I)
endfunction
Fbar=50;//lb/ft
```

```
Lbar=30;//ft
Ebar=1.5*(10^8);//lb/ft^2
Ibar=0.06;//ft^4
deltaF=2;//lb/ft
deltaL=0.1;//ft
deltaE=0.01*(10^8);//lb/ft^2
deltaI=0.0006;//ft^4
ybar=(Fbar*(Lbar^4))/(8*Ebar*Ibar);
function y=f1(F)
    y=(F*(Lbar^4))/(8*Ebar*Ibar)
endfunction
function y=f2(L)
    y=(Fbar*(L^4))/(8*Ebar*Ibar)
endfunction
function y=f3(E)
    y=(Fbar*(Lbar^4))/(8*E*Ibar)
endfunction
function y=f4(I)
    y=(Fbar*(Lbar^4))/(8*Ebar*I)
endfunction

deltay=abs(derivative(f1,Fbar))*deltaF+abs(derivative(f2,Lbar))*deltaL+abs(derivative(f3,Ebar))*deltaE+abs(derivative(f4,Ibar))*deltaI;

disp(ybar+deltay,"and",ybar-deltay,"The value of y is between:")
ymin=((Fbar-deltaF)*((Lbar-deltaL)^4))/(8*(Ebar+deltaE)*(Ibar+deltaI));
ymax=((Fbar+deltaF)*((Lbar+deltaL)^4))/(8*(Ebar-deltaE)*(Ibar-deltaI));
disp(ymin,"ymin is calculated at lower extremes of F, L, E, I values as =")
disp(ymax,"ymax is calculated at higher extremes of F, L, E, I values as =")
```

Scilab code Exa 4.7 Condition Number

```
clc;
clear;
function y=f(x)
    y=tan(x)
endfunction
x1bar=(%pi/2)+0.1*(%pi/2);
x2bar=(%pi/2)+0.01*(%pi/2);
//computing condition number for x1bar
condnum1=x1bar*derivative(f,x1bar)/f(x1bar);
disp(condnum1,"is:",x1bar,"The condition number of function for x=")
if abs(condnum1)>1 then disp(x1bar,"Function is ill-conditioned for x=")
end
//computing condition number for x2bar
condnum2=x2bar*derivative(f,x2bar)/f(x2bar);
disp(condnum2,"is:",x2bar,"The condition number of function for x=")
if abs(condnum2)>1 then disp(x2bar,"Function is ill-conditioned for x=")
end
```

Chapter 5

Bracketing Methods

Scilab code Exa 5.1 Graphical Approach

```
clc;
clear;
m=68.1;//kg
v=40;//m/s
t=10;//s
g=9.8;//m/s^2
function y=f(c)
    y=g*m*(1-exp(-c*t/m))/c - v;
endfunction
disp("For various values of c and f(c) is found as:"
  )
i=0;
for c=4:4:20
    i=i+1;
    bracket=[c f(c)];
    disp(bracket)
    fc(i)=f(c);
end
c=[4 8 12 16 20]
```

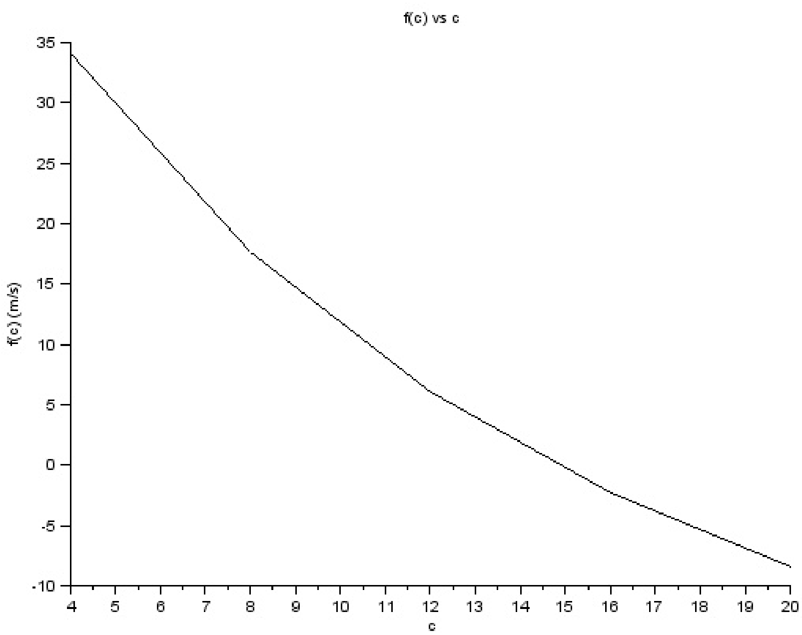

Figure 5.1: Graphical Approach

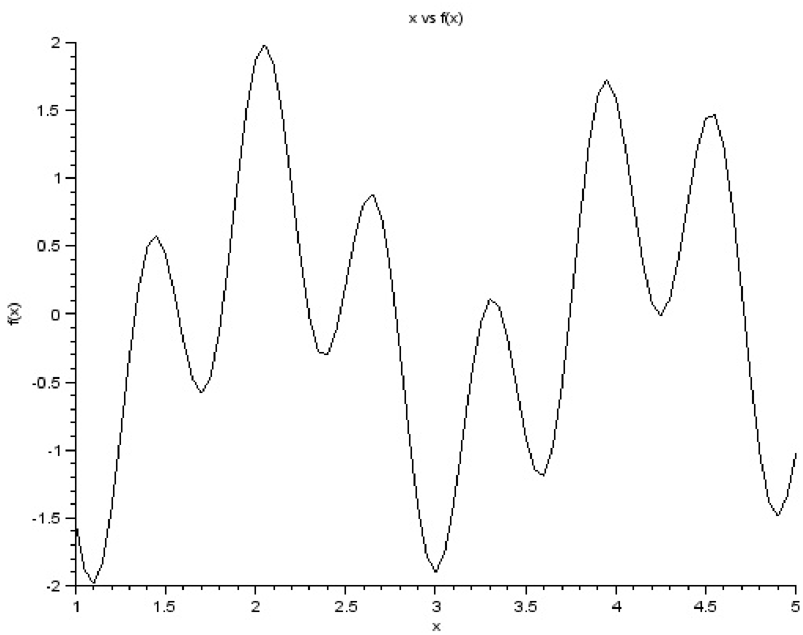

Figure 5.2: Computer Graphics to Locate Roots

```
plot2d(c,fc)
xtitle('f(c) vs c','c','f(c) (m/s)')
```

Scilab code Exa 5.2 Computer Graphics to Locate Roots

```
clc;
clear;
function y=f(x)
    y=sin(10*x)+cos(3*x);
endfunction
count=1;
```

```
for i=1:0.05:5
    val(count)=i;
    func(count)=f(i);
    count=count+1;
end
plot2d(val,func)
xtitle("x vs f(x)",'x','f(x)')
```

Scilab code Exa 5.3 Bisection

```
clc;
clear;
m=68.1;//kg
v=40;//m/s
t=10;//s
g=9.8;//m/s^2
function y=f(c)
    y=g*m*(1-exp(-c*t/m))/c - v;
endfunction
x1=12;
x2=16;
xt=14.7802;//true value
e=input("enter the tolerable true percent error=")
xr=(x1+x2)/2;
etemp=abs(xr-xt)/xt*100;//error
while etemp>e
    if f(x1)*f(xr)>0 then
        x1=xr;
        xr=(x1+x2)/2;
        etemp=abs(xr-xt)/xt*100;
    end
    if f(x1)*f(xr)<0 then
            x2=xr;
            xr=(x1+x2)/2;
            etemp=abs(xr-xt)/xt*100;
```

```
    end
    if f(x1)*f(xr)==0 then break;
    end
end

disp(xr,"The result is=")
```

Scilab code Exa 5.4 Error Estimates for Bisection

```
clc;
clear;
m=68.1;//kg
v=40;//m/s
t=10;//s
g=9.8;//m/s^2
function y=f(c)
    y=g*m*(1-exp(-c*t/m))/c - v;
endfunction
x1=12;
x2=16;
xt=14.7802;//true value
e=input("enter the tolerable approximate error=")
xr=(x1+x2)/2;
i=1;
et=abs(xr-xt)/xt*100;//error
disp(i,"Iteration:")
disp(x1,"xl:")
disp(x2,"xu:")
disp(xr,"xr:")
disp(et,"et(%):")
disp("−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−")
etemp=100;
while etemp>e
    if f(x1)*f(xr)>0 then
        x1=xr;
```

```
            xr=(x1+x2)/2;
            etemp=abs(xr-x1)/xr*100;
            et=abs(xr-xt)/xt*100;
        end
        if f(x1)*f(xr)<0 then
                x2=xr;
                xr=(x1+x2)/2;
                etemp=abs(xr-x2)/xr*100;
                et=abs(xr-xt)/xt*100;
        end
        if f(x1)*f(xr)==0 then break;
        end
i=i+1;
disp(i,"Iteration:")
disp(x1,"xl:")
disp(x2,"xu:")
disp(xr,"xr:")
disp(et,"et(%):")
disp(etemp,"ea(%)")
disp("-----------------------------------------")
end

disp(xr,"The result is=")
```

Scilab code Exa 5.5 False Position

```
clc;
clear;
m=68.1;//kg
v=40;//m/s
t=10;//s
g=9.8;//m/s^2
function y=f(c)
    y=g*m*(1-exp(-c*t/m))/c - v;
endfunction
```

```
x1=12;
x2=16;
xt=14.7802;//true value
e=input("enter the tolerable true percent error=")
xr=x1-(f(x1)*(x2-x1))/(f(x2)-f(x1));
etemp=abs(xr-xt)/xt*100;//error
while etemp>e
    if f(x1)*f(xr)>0 then
        x1=xr;
        xr=x1-(f(x1)*(x2-x1))/(f(x2)-f(x1));
        etemp=abs(xr-xt)/xt*100;
    end
    if f(x1)*f(xr)<0 then
            x2=xr;
            xr=x1-(f(x1)*(x2-x1))/(f(x2)-f(x1));
            etemp=abs(xr-xt)/xt*100;
    end
    if f(x1)*f(xr)==0 then break;
    end
end
disp(xr,"The result is=")
```

Scilab code Exa 5.6 Bracketing and False Position Methods

```
clc;
clear;
function y=f(x)
    y=x^10 - 1;
endfunction
x1=0;
x2=1.3;
xt=1;

//using bisection method
disp("BISECTION METHOD:")
```

```
xr=(x1+x2)/2;
et=abs(xr-xt)/xt*100;//error
disp(1,"Iteration:")
disp(x1,"xl:")
disp(x2,"xu:")
disp(xr,"xr:")
disp(et,"et(%):")
disp("--------------------------------------------")
for i=2:5
    if f(x1)*f(xr)>0 then
        x1=xr;
        xr=(x1+x2)/2;
        ea=abs(xr-x1)/xr*100;
        et=abs(xr-xt)/xt*100;
    else if f(x1)*f(xr)<0 then
            x2=xr;
            xr=(x1+x2)/2;
            ea=abs(xr-x2)/xr*100;
            et=abs(xr-xt)/xt*100;
        end
    end
    if f(x1)*f(xr)==0 then break;
    end
    disp(i,"Iteration:")
    disp(x1,"xl:")
    disp(x2,"xu:")
    disp(xr,"xr:")
    disp(et,"et(%):")
    disp(ea,"ea(%)")
    disp("--------------------------------------------")
end

//using false position method
disp("FALSE POSITION METHOD:")
x1=0;
x2=1.3;
xt=1;
xr=x1-(f(x1)*(x2-x1))/(f(x2)-f(x1));;
```

```
et=abs(xr-xt)/xt*100;//error
disp(1,"Iteration:")
disp(x1,"xl:")
disp(x2,"xu:")
disp(xr,"xr:")
disp(et,"et(%):")
disp("----------------------------------------------")
for i=2:5
    if f(x1)*f(xr)>0 then
        x1=xr;
        xr=x1-(f(x1)*(x2-x1))/(f(x2)-f(x1));
        ea=abs(xr-x1)/xr*100;
        et=abs(xr-xt)/xt*100;

    else if f(x1)*f(xr)<0 then
            x2=xr;
            xr=x1-(f(x1)*(x2-x1))/(f(x2)-f(x1));
            ea=abs(xr-x2)/xr*100;
            et=abs(xr-xt)/xt*100;
        end
    end
    if f(x1)*f(xr)==0 then break;
    end
    disp(i,"Iteration:")
    disp(x1,"xl:")
    disp(x2,"xu:")
    disp(xr,"xr:")
    disp(et,"et(%):")
    disp(ea,"ea(%)")
    disp("----------------------------------------------")
end
```

Chapter 6

Open Methods

Scilab code Exa 6.1 simple fixed point iteration

```
//clc()
//f(x) = exp(-x) - x;
//using simple fixed point iteration , Xi+1 = exp(-Xi
    )
x = 0;//initial guess
for i = 1:11
    if i == 1 then
        y(i) = x;
    else
         y(i) = exp(-y(i-1));
        e(i) = (y(i) - y(i-1)) * 100 / y(i);
    end
end
disp(y,"x = ")
disp("%",e,"e = ")
```

Scilab code Exa 6.2 The Two curve graphical method

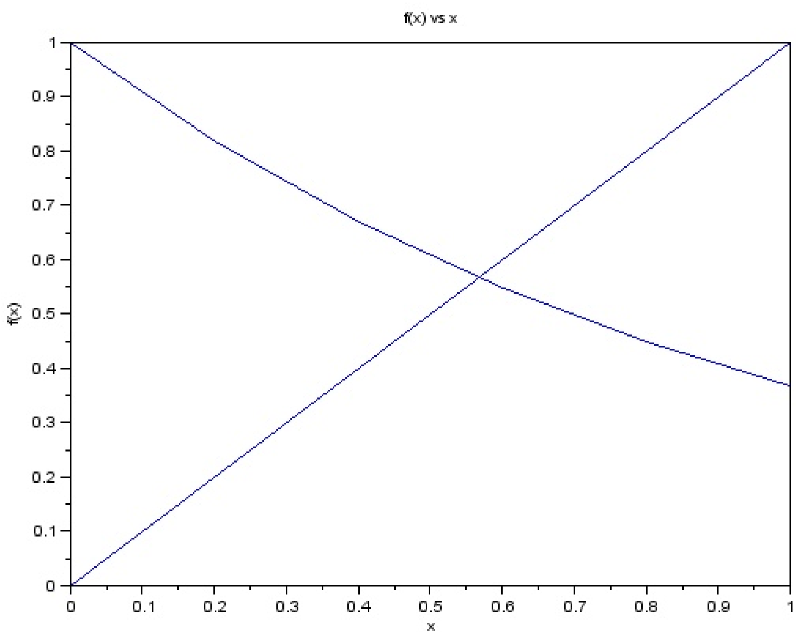

Figure 6.1: The Two curve graphical method

```
//clc()
//y1 = x
//y2 = exp(-x)
for i = 1:6
    if i == 1 then
        x(i) = 0;
       else   x(i) = x(i-1) + 0.2;
    end
    y1(i) = x(i);
    y2(i) = exp(-x(i));
end
disp(x,"x = ")
disp(y1,"y1 = ")
disp(y2,"y2 = ")
plot(x,y1);
plot(x,y2);
xtitle("f(x) vs x","x","f(x)")
// from the graph, we get
x7 = 5.7;
disp(x7, "answer using two curve graphical method =
    ");
```

Scilab code Exa 6.3 Newton Raphson Method

```
//clc()
//f(x) = exp(-x)-x
//f'(x) = -exp(-x)-1
for i = 1:5
    if i == 1 then
        x(i) = 0;
    else
         x(i) = x(i-1) - (exp(-x(i-1))-x(i-1))/(-exp
            (-x(i-1))-1);
         et(i) = (x(i) - x(i-1)) * 100 / x(i);
         x(i-1) = x(i);
```

```
    end
end
disp(x)
disp(et)
```

Scilab code Exa 6.4 Error analysis of Newton Raphson method

```
//clc()
//f(x) = exp(-x) - x
//f'(x) = -exp(-x) - 1
//f"(x) = exp(-x)
xr = 0.56714329;
//E(ti+1) = -f"(x)* E(ti) / 2 * f'(x)
Et0 = 0.56714329;
Et1 = -exp(-xr)* ((Et0)^2) / (2 * (-exp(-xr) - 1));
disp("which is close to the true error of 0.06714329
   ",Et1,"Et1 = ")
Et1true = 0.06714329;
Et2 =  -exp(-xr)* ((Et1true)^2) / (2 * (-exp(-xr) -
   1));
disp("which is close to the true error of 0.0008323"
   ,Et2,"Et2 = ")
Et2true = 0.0008323;
Et3 =  -exp(-xr)* ((Et2true)^2) / (2 * (-exp(-xr) -
   1));
disp("which is close to the true error",Et3,"Et3 = "
   )
Et4 =  -exp(-xr)* ((Et3)^2) / (2 * (-exp(-xr) - 1));
disp("which is close to the true error",Et4,"Et4 = "
   )
disp("Thus it illustratres that the error of newton
   raphson method for this case is proportional(by a
    factor of 0.18095) to the square of the error of
    the previous iteration")
```

Scilab code Exa 6.5 slowly converging function with Newton Raphson method

```
//clc()
z = 0.5;
//f(x) = x^10 - 1
//f'(x) = 10*x^9
for i = 1:40
    if i==1 then
        y(i) = z;
    else
        y(i) = y(i-1) - (y(i-1)^10 - 1)/(10*y(i-1)
           ^9)
    end
end
disp(y)
disp("Thus, after the first poor prediction, the
   technique is converging on to the true root of 1
   but at a very slow rate")
```

Scilab code Exa 6.6 The secant method

```
//clc()
funcprot(0)
//f(x) = exp(-x)-x
for i = 1:5
    if i==1 then
        x(i) = 0;
    else
        if i==2 then
            x(i) = 1;
        else
```

```
        x(i) =x(i-1) - (exp(-x(i-1))-x(i-1))*(x(i-2)
            - x(i-1))/((exp(-x(i-2))-x(i-2))-(exp(-x
           (i-1))-x(i-1)))
        er(i) = (0.56714329 - x(i)) * 100 /
           0.56714329;
        end
    end
end
disp(x(1:5),"x =")
disp(er(3:5),"et =")
```

Scilab code Exa 6.7 `secant and false position techniques`

```
//clc()
funcprot(0)
//f(x) = log(x)
disp("secant method")
for i = 1:4
    if i==1 then
        x(i) = 0.5;
    else
        if i==2 then
            x(i) = 5;
        else
        x(i) =x(i-1) - log(x(i-1)) * (x(i-2) - x(i
           -1))/(log(x(i-2)) - log(x(i-1)));
        end
    end
end
disp(x(1:4),"x =")
disp("thus, secant method is divergent")
disp("Now, False position method")
    xl = 0.5;
    xu = 5;
for i = 1:3
```

```
    m = log(xl);
    n = log(xu);
    xr = xu - n*(xl - xu)/(m - n);
    disp(xr,"xr = ")
    w = log(xr);
    if m*w < 0 then
        xu = xr;
    else
        xl = xr;
    end
end

disp("thus, false position method is convergent")
```

Scilab code Exa 6.8 Modified secant method

```
//clc()
del = 0.01;
z = 0.56714329
x1 = 1;
//f(x) = exp(-x) - x
disp(x1,"x1 = ")
for i = 1:4
    if i == 1 then
        x(i) = 1
    else
        w = x(i-1);
        m = exp(-x(i-1)) - x(i-1);
        x(i-1) = x(i-1)*(1+del);
        n = exp(-x(i-1)) - x(i-1);
        x(i) = w - (x(i-1)- w) * m/(n-m);
        em = (z - x(i))*100/z;
         disp(x(i),"x = ")
        disp("%",em," error = ")
    end
```

```
end
```

Scilab code Exa 6.9 `modified newton raphson method`

```
//clc()
//f(x) = x^3 - 5*x^2 + 7*x -3
//f'(x) = 3*x^2 - 10*x + 7
disp("standard Newton Raphson method")
for i = 1:7
    if i == 1 then
        x(i) = 0;
    else
         x(i) = x(i-1) - ((x(i-1))^3 - 5*(x(i-1))^2 + 7*x(i-1) -3)/(3*(x(i-1))^2 - 10*(x(i-1)) + 7);
         et(i) = (1 - x(i)) * 100 / 1;
         disp(x(i),"x = ")
         disp("%",et(i),"error = ")
         x(i-1) = x(i);
    end
end
disp("Modified Newton Raphson method")
//f"(x) = 6*x - 10
for i = 1:4
    if i == 1 then
        x(i) = 0;
    else
         x(i) = x(i-1) - ((x(i-1))^3 - 5*(x(i-1))^2 + 7*x(i-1) -3)*((3*(x(i-1))^2 - 10*(x(i-1)) + 7))/((3*(x(i-1))^2 - 10*(x(i-1)) + 7)^2 - ((x(i-1))^3 - 5*(x(i-1))^2 + 7*x(i-1) -3) * (6*x(i-1) - 10));
         et(i) = (1 - x(i)) * 100 / 1;
         disp(x(i),"x = ")
         disp("%",et(i),"error = ")
```

```
        x(i-1) = x(i);
    end
end
```

Scilab code Exa 6.10 `fixed point iteration for nonlinear system`

```
//clc()
//u(x,y) = x^2 + x*y - 10
//v(x,y) = y + 3*x*y^2 -57
for i = 1:4
    if i == 1 then
        x(i) = 1.5;
        y(i) = 3.5;
    else
        x(i) = sqrt(10 - (x(i-1))*(y(i-1)));
        y(i) = sqrt((57 - y(i-1))/(3*x(i)));
        disp(x(i),"x =")
        disp(y(i),"y =")
    end
end
disp("Thus the approaching to the true value 0f x =
   2 and y = 3")
```

Scilab code Exa 6.11 `Newton Raphson for a nonlinear Problem`

```
clc;
clear;
function z=u(x,y)
    z=x^2+x*y-10
endfunction
function z=v(x,y)
    z=y+3*x*y^2-57
endfunction
```

```
x=1.5;
y=3.5;
e=[100 100];
while  e(1)>0.0001 & e(2)>0.0001
    J=[2*x+y x; 3*y^2 1+6*x*y];
    deter=determ(J);
    u1=u(x,y);
    v1=v(x,y);
    x=x-((u1*J(2,2)-v1*J(1,2))/deter);
    y=y-((v1*J(1,1)-u1*J(2,1))/deter);
    e(1)=abs(2-x);
    e(2)=abs(3-y);
end
bracket=[x y];
disp(bracket)
```

Chapter 7

Roots of Polynomials

Scilab code Exa 7.1 Polynomial Deflation

```
clc;
clear;
function y=f(x)
    y=(x-4)*(x+6)
endfunction
n=2;
a(1)=-24;
a(2)=2;
a(3)=1;
t=4;
r=a(3);
a(3)=0;
for i=(n):-1:1
    s=a(i);
    a(i)=r;
    r=s+r*t;
end
disp("The quptient is a(1)+a(2)*x where :")
disp(a(1),"a(1)=")
disp(a(2),"a(2)=")
disp(r,"remainder=")
```

Scilab code Exa 7.2 Mullers Method

```
clc;
clear;
function y=f(x)
    y=x^3 - 13*x - 12
endfunction

x1t=-3;
x2t=-1;
x3t=4;
x0=4.5;
x1=5.5;
x2=5;
disp(0,"iteration:")
disp(x2,"xr:")
disp("-----------------------------------------------"
   )
for i=1:4

h0=x1-x0;
h1=x2-x1;
d0=(f(x1)-f(x0))/(x1-x0);
d1=(f(x2)-f(x1))/(x2-x1);
a=(d1-d0)/(h1+h0);
b=a*h1+d1;
c=f(x2);
d=(b^2 - 4*a*c)^0.5;
if abs(b+d)>abs(b-d) then x3=x2+((-2*c)/(b+d));
else x3=x2+((-2*c)/(b-d)); end
ea=abs(x3-x2)*100/x3;
x0=x1;
x1=x2;
x2=x3;
```

```
disp(i,"iteration:")
disp(x2,"xr:")
disp(ea,"ea(%):")
disp("------------------------------------------------------------"
    )
end
```

Scilab code Exa 7.3 Bairstows Method

```
clc;
clear;
function y=f(x)
    y=x^5-3.5*x^4+2.75*x^3+2.125*x^2-3.875*x+1.25
endfunction
r=-1;
s=-1;
es=1;//%
n=6;
count=1;
ear=100;
eas=100;
a=[1.25 -3.875 2.125 2.75 -3.5 1];
while (ear>es) and (eas>es)

    b(n)=a(n);
    b(n-1)=a(n-1)+r*b(n);
    for i=n-2:-1:1
        b(i)=a(i)+r*b(i+1)+s*b(i+2);
    end
    c(n)=b(n);
    c(n-1)=b(n-1)+r*c(n);
    for i=(n-2):-1:2
        c(i)=b(i)+r*c(i+1)+s*c(i+2);
    end
    //c(3)*dr+c(4)*ds=-b(2)
```

```
    //c(2)*dr+c(3)*ds=-b(1)
    ds=((-b(1))+(b(2)*c(2)/c(3)))/(c(3)-(c(4)*c(2)/c(3)));
    dr=(-b(2)-c(4)*ds)/c(3);
    r=r+dr;
    s=s+ds;
    ear=abs(dr/r)*100;
    eas=abs(ds/s)*100;
    disp(count,"Iteration:")
    disp(dr,"delata r:")
    disp(ds,"delata s:")
    disp(r,"r:")
    disp(s,"s:")
    disp(ear,"Error in r:")
    disp(eas,"Error in s:")
    disp("--------------------------------------------------------------------------")
    count=count+1;
end
x1=(r+(r^2 + 4*s)^0.5)/2;
x2=(r-(r^2 + 4*s)^0.5)/2;
bracket=[x1 x2];
disp(bracket,"The roots are:")
disp("x^3 - 4*x^2 + 5.25*x - 2.5","The quotient is:")
disp("--------------------------------------------------------------------------")
```

Scilab code Exa 7.4 Locate single root

```
clc;
clear;
function y=f(x)
```

```
    y=x-cos(x)
endfunction
x1=0;
if f(x1)<0 then
    x2=x1+0.001;
    while f(x2)<0
        x2=x2+0.001;
    end
elseif x2=x1+0.001;
    while f(x2)>0
        x2=x2+0.001;
    end
else disp(x1,"The root is=")
end
x=x2-(x2-x1)*f(x2)/(f(x2)-f(x1));
disp(x,"The root is=")
```

Scilab code Exa 7.5 Solving nonlinear system

```
clc;
clear;
function z=u(x,y)
    z=x^2+x*y-10
endfunction
function z=v(x,y)
    z=y+3*x*y^2-57
endfunction
x=1;
y=3.5;
while u(x,y)~=v(x,y)
    x=x+1;
    y=y-0.5;
end
disp(x,"x=")
disp(y,"y=")
```

Scilab code Exa 7.6 Root Location

```
clc;
clear;
x=poly(0,'s');
p=x^10 -1;
disp("The roots of the polynomial are:")
disp(roots(p))
```

Scilab code Exa 7.7 Roots of Polynomials

```
clc;
clear;
x=poly(0,'s');
p=x^5 - 3.5*x^4 +2.75*x^3 +2.125*x^2 - 3.875*x +
   1.25;
disp("The roots of the polynomial are:")
disp(roots(p))
```

Scilab code Exa 7.8 Root Location

```
clc;
clear;
function y=f(x)
    y=x-cos(x)
endfunction
x1=0;
if f(x1)<0 then
    x2=x1+0.00001;
```

```
    while f(x2)<0
        x2=x2+0.00001;
    end
elseif x2=x1+0.00001;
    while f(x2)>0
        x2=x2+0.00001;
    end
else disp(x1,"The root is=")
end
x=x2-(x2-x1)*f(x2)/(f(x2)-f(x1));
disp(x,"The root is=")
```

Chapter 9

Gauss Elimination

Scilab code Exa 9.1 Graphical Method for two Equations

```
clc;
clear;
//the equations are:
//3*x1 + 2*x2=18 and -x1 + 2*x2=2

//equation 1 becomes,
//x2=-(3/2)*x1 + 9
//equation 2 becomes,
//x2=-(1/2)*x1 + 1

//plotting equation 1
for x1=1:6
    x2(x1)=-(3/2)*x1 + 9;
end
x1=[1 2 3 4 5 6];
//plotting equation 2
for x3=1:6
    x4(x3)=(1/2)*x3 + 1;
end
```

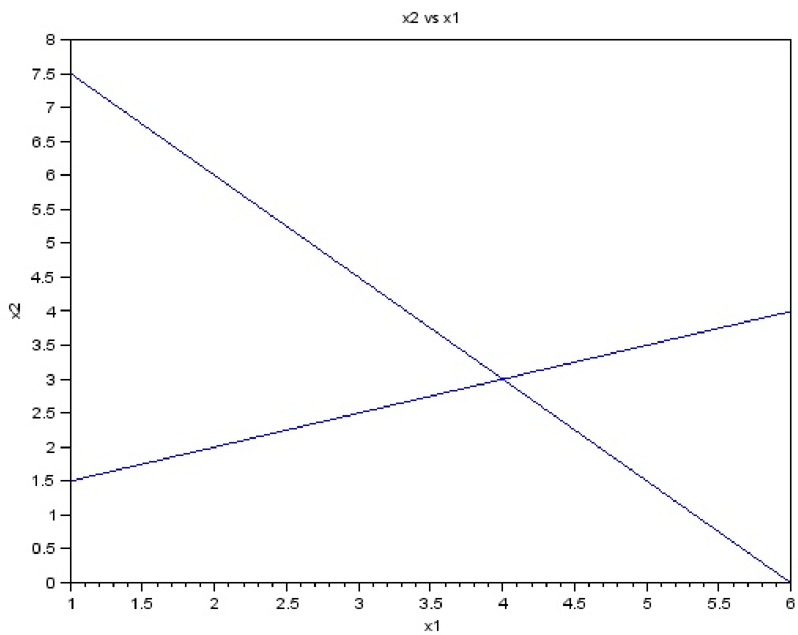

Figure 9.1: Graphical Method for two Equations

```
x3=[1 2 3 4 5 6];
plot(x1,x2)
plot(x3,x4)
xtitle("x2 vs x1","x1","x2")
//the lines meet at x1=4 amd x2=3
disp(3,"x2=","and",4,"x1=","The lines meet at=")
```

Scilab code Exa 9.2 Deterinants

```
clc;
clear;
//For fig9.1
a=[3 2;-1 2];
disp(determ(a),"The value of determinant for system repesented in fig 9.1 =")
//For fig9.2 (a)
a=[-0.5 1;-0.5 1];
disp(determ(a),"The value of determinant for system repesented in fig 9.2 (a) =")
//For fig9.2 (b)
a=[-0.5 1;-1 2];
disp(determ(a),"The value of determinant for system repesented in fig 9.2 (b) =")
//For fig9.2 (c)
a=[-0.5 1;-2.3/5 1];
disp(determ(a),"The value of determinant for system repesented in fig 9.2 (c) =")
```

Scilab code Exa 9.3 Cramers Rule

```
clc;
clear;
//the matrix or the system
```

```
b1=-0.01;
b2=0.67;
b3=-0.44;
a=[0.3 0.52 1;0.5 1 1.9;0.1 0.3 0.5];
a1=[a(2,2) a(2,3);a(3,2) a(3,3)];
A1=determ(a1);
a2=[a(2,1) a(2,3);a(3,1) a(3,3)];
A2=determ(a2);
a3=[a(2,1) a(2,2);a(3,1) a(3,2)];
A3=determ(a3);
D=a(1,1)*A1-a(1,2)*A2+a(1,3)*A3;
p=[b1 0.52 1;b2 1 1.9;b3 0.3 0.5];
q=[0.3 b1 1;0.5 b2 1.9;0.1 b3 0.5];
r=[0.3 0.52 b1;0.5 1 b2;0.1 0.3 b3];
x1=det(p)/D;
x2=det(q)/D;
x3=det(r)/D;
disp("The values are:")
disp(x1,"x1=")
disp(x2,"x2=")
disp(x3,"x3=")
```

Scilab code Exa 9.4 Elimination of Unknowns

```
clc;
clear;
//the equations are:
//3*x1+2*x2=18
//-x1+2*x2=2
a11=3;
a12=2;
b1=18;
a21=-1;
a22=2;
b2=2;
```

```
x1=(b1*a22-a12*b2)/(a11*a22-a12*a21);
x2=(b2*a11-a21*b1)/(a11*a22-a12*a21);
disp(x1,"x1=")
disp(x2,"x2=")
```

Scilab code Exa 9.5 Naive Gauss Elimination

```
clc;
clear;
n=3;
b(1)=7.85;
b(2)=-19.3;
b(3)=71.4;
a=[3 -0.1 -0.2;0.1 7 -0.3;0.3 -0.2 10];
for k=1:n-1
    for i=k+1:n
        fact=a(i,k)/a(k,k);
        for j=k+1:n
            a(i,j)=a(i,j)-fact*(a(k,j));
        end
        b(i)=b(i)-fact*b(k);
    end
end
x(n)=b(n)/a(n,n);
for i=n-1:-1:1
    s=b(i);
    for j=i+1:n
        s=s-a(i,j)*x(j)
    end
    x(i)=b(i)/a(i,i);
end
disp(x(1),"x1=")
disp(x(2),"x2=")
disp(x(3),"x3=")
```

Scilab code Exa 9.6 ill conditioned systems

```
clc;
clear;
a11=1;
a12=2;
b1=10;
a21=1.1;
a22=2;
b2=10.4;
x1=(b1*a22-a12*b2)/(a11*a22-a12*a21);
x2=(b2*a11-a21*b1)/(a11*a22-a12*a21);
disp("For the original system:")
disp(x1,"x1=")
disp(x2,"x2=")
a21=1.05;
x1=(b1*a22-a12*b2)/(a11*a22-a12*a21);
x2=(b2*a11-a21*b1)/(a11*a22-a12*a21);
disp("For the new system:")
disp(x1,"x1=")
disp(x2,"x2=")
```

Scilab code Exa 9.7 Effect of Scale on Determinant

```
clc;
clear;
//part a
a=[3 2;-1 2];
b1=18;
b2=2;
disp(determ(a),"The determinant for part(a)=")
//part b
```

```
a=[1 2;1.1 2];
b1=10;
b2=10.4;
disp(determ(a),"The determinant for part(b)=")
//part c
a1=a*10;
b1=100;
b2=104;
disp(determ(a1),"The determinant for part(c)=")
```

Scilab code Exa 9.8 Scaling

```
clc;
clear;
//part a
a=[1 0.667;-0.5 1];
b1=6;
b2=1;
disp(determ(a),"The determinant for part(a)=")
//part b
a=[0.5 1;0.55 1];
b1=5;
b2=5.2;
disp(determ(a),"The determinant for part(b)=")
//part c
b1=5;
b2=5.2;
disp(determ(a),"The determinant for part(c)=")
```

Scilab code Exa 9.9 Partial Pivoting

```
//clc()
//0.0003*x1 + 3*x2 = 2.0001
```

```
//1*x1 + 1*x2 = 1
a11 = 0.000;
//multiplying first equation by 1/0.0003, we get, x1 + 10000*x2 = 6667
x2 = (6667-1)/(10000-1);
x1 = 6667 - 10000*x2;
disp(x2,"x2 = ")
disp(x1,"x1 = ")
disp("The error varies depending on the no. of significant figures used")
```

Scilab code Exa 9.10 `Effect of scaling on Pivoting and round off`

```
//clc()
//2*x1 + 10000*x2 = 10000
//x1 + x2 = 2
x1 = 1;
x2 = 1;
disp("without scaling, applying forward elimination")
//x1 is too small and can be neglected
x21 = 10000/10000;
x11 = 0;
e1 = (x1 - x11)*100/x1;
disp(x21,"x2 = ")
disp(x11,"x1 = ")
disp(e1,"error for x1 = ")
disp("with scaling")
//0.00002*x1 + x2 = 1
//now x1 is neglected because of the co efficient
x22 = 1;
x12 = 2 - x1;
disp(x12,"x1 = ")
disp(x22,"x2 = ")
//using original co efficient
```

```
//x1 can be neglected
disp("pivot and retaining original coefficients")
x22 = 10000/10000;
x12 = 2 - x1;
disp(x12,"x1 = ")
disp(x22,"x2 = ")
```

Scilab code Exa 9.11 Solution of Linear Algebraic Equations

```
clc;
clear;
a=[70 1 0;60 -1 1;40 0 -1];
b=[636;518;307];
x=abs(linsolve(a,b));
disp("m/s^2",x(1,1),"a=")
disp("N",x(2,1),"T=")
disp("N",x(3,1),"R=")
```

Scilab code Exa 9.12 Gauss Jordan method

```
//clc()
//3*x1 - 0.1*x2 - 0.2*x3 = 7.85
//0.1*x1 + 7*x2 - 0.3*x3 = -19.3
//0.3*x1 - 0.2*x2 + 10*x3 = 71.4
// this can be written in matrix form as
A =
    [3,-0.1,-0.2,7.85;0.1,7,-0.3,-19.3;0.3,-0.2,10,71.4];

disp(A,"Equation in matrix form can be written as")
X = A(1,:) / det(A(1,1));
Y = A(2,:) - 0.1*X;
Z = A(3,:) - 0.3*X;
Y = Y/det(Y(1,2));
```

```
X = X - Y * det(X(1,2));
Z = Z - Y * det(Z(1,2));
Z = Z/det(Z(1,3));
X = X - Z*det(X(1,3));
Y = Y - Z*det(Y(1,3));
A = [X;Y;Z];
disp(A,"final matrix = ")
disp(det(A(1,4)),"x1 = ")
disp(det(A(2,4)),"x2 = ")
disp(det(A(3,4)),"x3 = ")
```

Chapter 10

LU Decomposition and matrix inverse

Scilab code Exa 10.1 `LU decomposition with gauss elimination`

```
//clc()
A = [3,-0.1,-0.2;0.1,7,-0.3;0.3,-0.2,10];
U = A;
disp(A,"A =")
m = det(U(1,1));
n = det(U(2,1));
a = n/m;
U(2,:) = U(2,:) - U(1,:) / (m/n);
n = det(U(3,1));
b = n/m;
U(3,:) = U(3,:) - U(1,:) / (m/n);
m = det(U(2,2));
n = det(U(3,2));
c = n/m;
U(3,:) = U(3,:) - U(2,:) / (m/n);
disp(U,"U = ")
L = [1,0,0;a,1,0;b,c,1];
disp(L,"L calculated based on gauss elimination
    method = ")
```

Scilab code Exa 10.2 `The substitution steps`

```
//clc()
A = [3,-0.1,-0.2;0.1,7,-0.3;0.3,-0.2,10];
B = [7.85;-19.3;71.4];
X = inv(A) * B;
disp(X,"X = ")
```

Scilab code Exa 10.3 `Matrix inversion`

```
//clc()
A = [3,-0.1,-0.2;0.1,7,-0.3;0.3,-0.2,10];
//B = inv(A)
L = [1,0,0;0.033333,1,0;0.1,-0.02713,1];
U = [3,-0.1,-0.2;0,7.0033,-0.293333;0,0,10.012];
for i =1:3
    if i==1 then
        m = [1;0;0];
    else
        if i==2 then
            m = [0;1;0];
        else
            m = [0;0;1];
        end
    end
    d = inv(L) * m;
    x = inv(U) * d;
    B(:,i) = x
end
disp(B)
```

Scilab code Exa 10.4 Matrix condition evaluation

```
//clc()
A = [1,1/2,1/3;1/2,1/3,1/4;1/3,1/4,1/5];
n = det(A(2,1));
A(2,:) = A(2,:)/n;
n = det(A(3,1));
A(3,:) = A(3,:)/n;
B = inv(A);
disp(A,"A = ")
for j = 1:3
    a = 0;
for i = 1:3
    m(i) = det(A(j,i));
    su(j) = a + m(i);
    a = su(j);
end
end
if su(1) < su(2) then
    if su(2) < su(3) then
        z = su(3);
    else
        z = su(2);
    end
else
     if su(1) < su(3) then
        z = su(3);
    else
        z = su(1);
    end
end
for j = 1:3
    a = 0;
for i = 1:3
```

```
    m(i) = det(B(j,i));
    sm(j) = a + abs(m(i));
    a = sm(j);
end
end
if sm(1) < sm(2) then
    if sm(2) < sm(3) then
        y = sm(3);
    else
        y = sm(2);
    end
else
     if sm(1) < sm(3) then
        y = sm(3);
    else
        y = sm(1);
    end
end
C = z*y;
disp(C,"Condition number for the matrix =")
```

Chapter 11

Special Matrices and gauss seidel

Scilab code Exa 11.1 Tridiagonal solution with Thomas algorithm

```
//clc()
A = [2.04,-1,0,0;-1,2.04,-1,0;0,-1,2.04,-1;0,0,-1,2.04];
B = [40.8;0.8;0.8;200.8];
g = det(A(1,2));
f1 =det(A(1,1));
e2 = det(A(2,1))/f1;
f2 = det(A(1,1)) - e2 * det(A(2,1));
e3 = det(A(2,1))/f2;
f3 = det(A(1,1)) - e3 * det(A(2,1));
e4 = det(A(2,1))/f3;
f4 = det(A(1,1)) - e4 * det(A(2,1));
M = [f1,g,0,0;e2,f2,g,0;0,e3,f3,g;0,0,e4,f4];
L = [1,0,0,0;det(M(2,1)),1,0,0;0,det(M(3,2)),1,0;0,0,det(M(4,3)),1];
U = [det(M(1,1)),g,0,0;0,det(M(2,2)),g,0;0,0,det(M(3,3)),g;0,0,0,det(M(4,4))];
r1 = det(B(1,1));
```

```
r2 = det(B(2,1)) - e2*det(B(1,1));
r3 = det(B(3,1)) - e3*r2;
r4= det(B(4,1)) - e4*r3;
N = [r1;r2;r3;r4];
T4 = r4/det(U(4,4));
T3 = (r3 - g*T4)/det(U(3,3));
T2 = (r2 - g*T3)/det(U(2,2));
T1 = (r1 - g*T2)/det(U(1,1));
disp(T1,"T1 = ")
disp(T2,"T2 = ")
disp(T3,"T3 = ")
disp(T4,"T4 = ")
```

Scilab code Exa 11.2 Cholesky Decomposition

```
//clc()
A = [6,15,55;15,55,225;55,225,979];
sl = 0
l11 = (det(A(1,1)))^(1/2);
//for second row
l21 = (det(A(2,1)))/l11;
l22 = (det(A(2,2)) - l21^2)^(0.5);
//for third row
l31 = (det(A(3,1)))/l11;
l32 = (det(A(3,2)) - l21*l31)/l22;
l33 = (det(A(3,3)) - l31^2 - l32^2)^(0.5);
L = [l11,0,0;l21,l22,0;l31,l32,l33];
disp(L,"L = ")
```

Scilab code Exa 11.3 Gauss Seidel method

```
//clc()
//3x - 0.1y - 0.2z = 7.85
```

```
//0.1x + 7y − 0.3z = −19.3
//0.3x − 0.2y + 10z = 71.4
Y = 0;
Z = 0;
for i = 1:2
    x(i) = (7.85 +0.1*Y+0.2*Z)/3;
    X = x(i);
    y(i) = (-19.3 - 0.1*X +0.3*Z)/7;
    Y = y(i);
    z(i) = (71.4 - 0.3*X+0.2*Y)/10;
    Z = z(i);
    if i==2 then
        ex = (x(i) - x(i-1))*100/x(i);
        ey = (y(i) - y(i-1))*100/y(i);
        ez = (z(i) - z(i-1))*100/z(i);
    end
end
disp(x(1:2),"x through two iterations =")
disp(y(1:2),"y through two iterations =")
disp(z(1:2),"z through two iterations =")
disp("%",ex,"error of x = ")
disp("%",ey,"error of y = ")
disp("%",ez,"error of z = ")
```

Scilab code Exa 11.4 Linear systems

```
//clc()
A = [1,0.5,1/3;1,2/3,1/2;1,3/4,3/5];
B = [1.833333;2.166667;2.35];
U = inv(A);
X = U*B;
x = det(X(1,1));
y = det(X(2,1));
z = det(X(3,1));
disp(x,"x = ")
```

```
disp(y,"y = ")
disp(z,"z = ")
```

Scilab code Exa 11.5 `Manipulate linear algebraic equations`

```
//clc()
A = [1,0.5,1/3;1,2/3,1/2;1,3/4,3/5];
B = [1.833333;2.166667;2.35];
U = inv(A);
X = U*B;
x = det(X(1,1));
y = det(X(2,1));
z = det(X(3,1));
disp(x,"x = ")
disp(y,"y = ")
disp(z,"z = ")
```

Scilab code Exa 11.6 `Analyze and solve Hilbert matrix`

```
//clc()
A = [1,0.5,1/3;1/2,1/3,1/4;1/3,1/4,1/5];
B = [1.833333;1.083333;0.783333];
U = inv(A);
X = U*B;
x = det(X(1,1));
y = det(X(2,1));
z = det(X(3,1));
disp(x,"x = ")
disp(y,"y = ")
disp(z,"z = ")
```

Chapter 13

One dimensional unconstrained optimization

Scilab code Exa 13.1 Golden section method

```
//clc()
//f(x) = 2sinx - x^2/10
xl(1) = 0;
xu(1) = 4;
for i = 1:10
    d(i) = ((5)^(0.5) - 1)*(xu(i) - xl(i))/2;
    x1(i) = xl(i) + d(i);
    x2(i) = xu(i) - d(i);
    m(i) = 2*sin(x1(i)) - (x1(i)^2)/10;
    n(i) = 2*sin(x2(i)) - (x2(i)^2)/10;
    if n(i) > m(i) then
        xu(i+1) = x1(i);
        xl(i+1) = xl(i);
    else
        xl(i+1) = x2(i);
        xu(i+1) = xu(i);
    end
end
disp(xl,"xl = ")
```

```
disp(x2,"x2 = ")
disp(x1,"x1 = ")
disp(xu,"xu = ")
```

Scilab code Exa 13.2 Quadratic interpolation

```
//clc()
//f(x) = 2sinx - x^2/10
x0(1) = 0;
x1(1) = 1
x2(1) = 4;
for i = 1:6
    m(i) = 2*sin(x0(i)) - (x0(i)^2)/10;
    n(i) = 2*sin(x1(i)) - (x1(i)^2)/10;
    r(i) = 2*sin(x2(i)) - (x2(i)^2)/10;
    x3(i) = ((m(i)*(x1(i) ^ 2 -x2(i) ^ 2)) + (n(i)*(
       x2(i) ^ 2 -x0(i) ^ 2)) + (r(i)*(x0(i) ^ 2 -x1
       (i) ^ 2)))/((2*m(i)*(x1(i) -x2(i)))+(2*n(i)*(
       x2(i) -x0(i)))+(2*r(i)*(x0(i) -x1(i))));
    s(i) = 2*sin(x3(i)) - (x3(i)^2)/10;
    if x1(i) > x3(i) then
        if n(i)<s(i) then
            x0(i+1) = x0(i);
            x1(i+1) = x3(i);
            x2(i+1) = x1(i);
        else
            x0(i+1) = x1(i);
            x1(i+1) = x3(i);
            x2(i+1) = x2(i);
        end
    else
        if n(i)>s(i) then
            x0(i+1) = x0(i);
            x1(i+1) = x3(i);
            x2(i+1) = x1(i);
```

```
            else
                x0(i+1) = x1(i);
                x1(i+1) = x3(i);
                x2(i+1) = x2(i);
            end
        end
end
disp(x0(1:6),"x0 = ")
disp(x1(1:6),"x1 = ")
disp(x3(1:6),"x3 = ")
disp(x2(1:6),"x2 = ")
```

Scilab code Exa 13.3 `Newtons method`

```
//clc()
//f(x) = 2sinx - x^2/10
x(1) = 2.5;
//f'(x) = 2cosx - x/5
//f"(x) = -2sinx - 1/5
for i = 2:10
  x(i) = x(i-1) - (2*cos(x(i-1)) - x(i-1)/5)/(-2*sin
     (x(i-1)) - 1/5);
end
disp(x,"x = ")
```

Chapter 14

Multidimensional Unconstrained Optimization

Scilab code Exa 14.1 `Random Search Method`

```
clc;
clear;
function z=f(x,y)
    z=y-x-(2*(x^2))-(2*x*y)-(y^2);
endfunction
x1=-2;
x2=2;
y1=1;
y2=3;
fmax=-1*10^(-15);
n=10000;
for j=1:n
    r=rand(1,2);
    x=x1+(x2-x1)*r(1,1);
    y=y1+(y2-y1)*r(1,2);
    fn=f(x,y);
    if fn>fmax then
        fmax=fn;
        xmax=x;
```

```
        ymax=y;
    end
    if modulo(j,1000)==0 then

    disp(j,"Iteration:")
    disp(x,"x:")
    disp(y,"y:")
    disp(fn,"function value:")
    disp("———————————————————————————————————————
      ")
    end
end
```

Scilab code Exa 14.2 Path of Steepest Descent

```
clc;
clear;
function z=f(x,y)
    z=x*y*y
endfunction
p1=[2 2];
elevation=f(p1(1),p1(2));
dfx=p1(1)*p1(1);
dfy=2*p1(1)*p1(2);
theta=atan(dfy/dfx);
slope=(dfx^2 + dfy^2)^0.5;
disp(elevation,"Elevation:")
disp(theta,"Theta:")
disp(slope,"slope:")
```

Scilab code Exa 14.3 1 D function along Gradient

```
clc;
```

```
clear;
function z=f(x,y)
    z=2*x*y + 2*x - x^2 - 2*y^2
endfunction
x=-1;
y=1;
dfx=2*y+2-2*x;
dfy=2*x-4*y;
//the function can thus be expressed along h axis as
//f((x+dfx*h),(y+dfy*h))
disp("180*h^2 + 72*h - 7","The final equation is=")
```

Scilab code Exa 14.4 Optimal Steepest Descent

```
clc;
clear;
function z=f(x,y)
    z=2*x*y + 2*x - x^2 - 2*y^2
endfunction
x=-1;
y=1;
d2fx=-2;
d2fy=-4;
d2fxy=2;

modH=d2fx*d2fy-(d2fxy)^2;

for i=1:25
dfx=2*y+2-2*x;
dfy=2*x - 4*y;
//the function can thus be expressed along h axis as
//f((x+dfx*h),(y+dfy*h))
function d=g(h)
    d=2*(x+dfx*h)*(y+dfy*h) + 2*(x+dfx*h) - (x+dfx*h
       )^2 - 2*(y+dfy*h)^2
```

```
endfunction
//2*dfx*(y+dfy*h)+2*dfy*(x+dfx*h)+2*dfx-2*(x+dfx*h)*
    dfx-4*(y+dfy*h)*dfy=g'(h)=0
//2*dfx*y + 2*dfx*dfy*h + 2*dfy*x + 2*dfy*dfx*h + 2*
    dfx - 2*x*dfx - 2*dfx*dfx*h - 4*y*dfy - 4*dfy*dfy
    *h=0
//h(2*dfx*dfy+2*dfy*dfx-2*dfx*dfx-4*dfy*dfy)=-(2*dfx
    *y+2*dfy*x-2*x*dfx-4*y*dfy)
h=(2*dfx*y+2*dfy*x-2*x*dfx-4*y*dfy+2*dfx)/(-1*(2*dfx
    *dfy+2*dfy*dfx-2*dfx*dfx-4*dfy*dfy));
x=x+dfx*h;
y=y+dfy*h;
end
disp([x y],"The final values are:")
```

Chapter 15

Constrained Optimization

Scilab code Exa 15.1 Setting up LP problem

```
clc;
clear;
regular=[7 10 9 150];
premium=[11 8 6 175];
res_avail=[77 80];
//total profit(to be maximized)=z=150*x1+175*x2
//total gas used=7*x1+11*x2 (has to be less than 77
   m^3/week)
//similarly other constraints are developed
disp("Maximize z=150*x1+175*x2")
disp("subject to")
disp("7*x1+11*x2<=77 (Material constraint)")
disp("10*x1+8*x2<=80 (Time constraint)")
disp("x1<=9 (Regular storage constraint)")
disp("x2<=6 (Premium storage constraint)")
disp("x1,x2>=0 (Positivity constraint)")
```

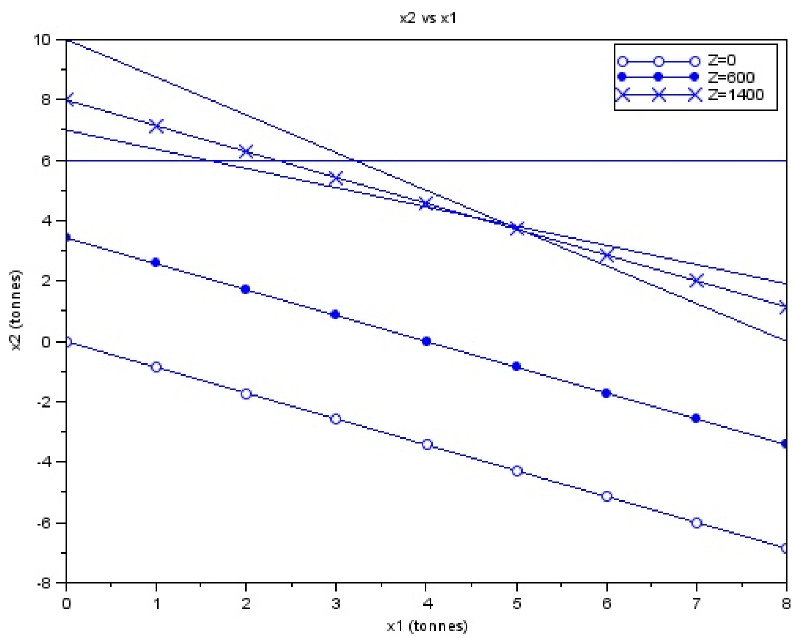

Figure 15.1: Graphical Solution

Scilab code Exa 15.2 Graphical Solution

```
clc;
clear;
for x1=0:8
    x21(x1+1)=-(7/11)*x1+7;
    x22(x1+1)=(80-10*x1)/8;
    x23(x1+1)=6;
    x24(x1+1)=-150*x1/175;
    x25(x1+1)=(600-150*x1)/175;
    x26(x1+1)=(1400-150*x1)/175;
end
x1=0:8;

plot(x1,x24,'o-')
plot(x1,x25,'.-')
plot(x1,x26,'x-')
h1=legend(['Z=0';'Z=600';'Z=1400'])
plot(x1,x21);
plot(x1,x22);
plot(x1,x23);
xtitle('x2 vs x1','x1 (tonnes)','x2 (tonnes)')
```

Scilab code Exa 15.3 Linear Programming Problem

```
clc;
clear;
x1=[4.888889 3.888889];
x2=[7 11];
x3=[10 8];
x4=[150 175];
x5=[77 80 9 6];
profit=[x1(1)*x4(1) x1(2)*x4(2)];
total=[x1(1)*x3(1)+x1(2)*x3(2) x1(1)*x3(1)+x1(2)*x3
   (2) x1(1) x1(2) profit(1)+profit(2)];
```

```
e=1000;

while e>total(5)
    if total(1)<=x5(1) then
        if total(2)<=x5(2) then
            if total(3)<=x5(3) then
                if total(4)<=x5(4) then
                    l=1;
                end
            end
        end
    end
    if l==1 then
        x1(1)=x1(1)+4.888889;
        x1(2)=x1(2)+3.888889;
        profit=[x1(1)*x4(1) x1(2)*x4(2)];
        total(5)=profit(1)+profit(2);
    end
end
disp(total(5),"The maximized profit is=")
```

Scilab code Exa 15.4 Nonlinear constrained optimization

```
clc;
clear;
Mt=2000;//kg
g=9.8;//m/s^2
c0=200;//$
c1=56;//$/m
c2=0.1;//$/m^2
vc=20;//m/s
kc=3;//kg/(s*m^2)
z0=500;//m
t=27;
r=2.943652;
```

```
n=6;
A=2*%pi*r*r;
l=(2^0.5)*r;
c=3*A;
m=Mt/n;
function y=f(t)
    y=(z0+g*m*m/(c*c)*(1-exp(-c*t/m)))*c/(g*m);
endfunction

    while abs(f(t)-t)>0.00001
        t=t+0.00001;
    end
v=g*m*(1-exp(-c*t/m))/c;
disp(v,"The final value of velocity=")
disp(n,"The final no. of load parcels=")
disp("m",r,"The chute radius=")
disp((c0+c1*l+c2*A*A)*n,"The minimum cost($)=")
```

Scilab code Exa 15.5 One dimensional Optimization

```
clc;
clear;
function y=fx(x)
    y=-(2*sin(x))+x^2/10
endfunction
x=fminsearch(fx,0)
disp("After maximization:")
disp(x,"x=")
disp(fx(x),"f(x)=")
```

Scilab code Exa 15.6 Multidimensional Optimization

```
clc;
```

```
clear;
function f=fx(x)
    f=-(2*x(1)*x(2)+2*x(1)-x(1)^2-2*x(2)^2)
endfunction
x=fminsearch(fx,[-1 1])
disp("After maximization:")
disp(x,"x=")
disp(fx(x),"f(x)=")
```

Scilab code Exa 15.7 Locate Single Optimum

```
clc;
clear;
function y=fx(x)
    y=-(2*sin(x)-x^2/10)
endfunction
x=fminsearch(fx,0)
disp("After maximization:")
disp(x,"x=")
disp(fx(x),"f(x)=")
```

Chapter 17

Least squares regression

Scilab code Exa 17.1 `Linear regression`

```
//clc()
x = [1,2,3,4,5,6,7];
y = [0.5,2.5,2,4,3.5,6,5.5];
n = 7;
s = 0;
xsq = 0;
xsum = 0;
ysum = 0;
for i = 1:7
    s = s + (det(x(1,i)))*(det(y(1,i)));
    xsq = xsq + (det(x(1,i))^2);
    xsum = xsum + det(x(1,i));
    ysum = ysum + det(y(1,i));
end
disp(s,"sum of product of x and y =")
disp(xsq,"sum of square of x = ")
disp(xsum,"sum of all the x = ")
disp(ysum,"sum of all the y = ")
a = xsum/n;
b = ysum/n;
a1 = (n*s - xsum*ysum)/(n*xsq -xsum^2);
```

```
a0 = b - a*a1;
disp(a1,"a1 = ")
disp(a0,"a0 = ")
disp("The equation of the line obtained is y = a0 +
   a1*x")
```

Scilab code Exa 17.2 Estimation of errors for the linear least square fit

```
//clc()
x = [1,2,3,4,5,6,7];
y = [0.5,2.5,2,4,3.5,6,5.5];
n = 7;
s = 0;
ssum = 0;
xsq = 0;
xsum = 0;
ysum = 0;
msum = 0;
for i = 1:7
    s = s + (det(x(1,i)))*(det(y(1,i)));
    xsq = xsq + (det(x(1,i))^2);
    xsum = xsum + det(x(1,i));
    ysum = ysum + det(y(1,i));
end
a = xsum/n;
b = ysum/n;
a1 = (n*s - xsum*ysum)/(n*xsq -xsum^2);
a0 = b - a*a1;
for i = 1:7
    m(i) = (det(y(1,i)) - ysum/7)^2;
    msum = msum +m(i);
    si(i) = (det(y(1,i)) - a0 - a1*det(x(1,i)))^2;
    ssum = ssum + si(i);
end
disp(ysum,"sum of all y =")
```

```
disp(m,"(yi - yavg)^2 = ")
disp(msum,"total (yi - yavg)^2 = ")
disp(si,"(yi - a0 - a1*x)^2 = ")
disp(ssum,"total (yi - a0 - a1*x)^2 = ")
sy = (msum / (n-1))^(0.5);
sxy = (ssum/(n-2))^(0.5);
disp(sy,"sy = ")
disp(sxy,"sxy = ")
r2 = (msum - ssum)/(msum);
r = r2^0.5;
disp(r,"r = ")
disp("The result indicate that 86.8 percent of the
   original uncertainty has been explained by linear
    model")
```

Scilab code Exa 17.3.a linear regression using computer

```
//clc()
s = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15];
v =
   [10,16.3,23,27.5,31,35.6,39,41.5,42.9,45,46,45.5,46,49,50];

g = 9.8//m/s^2
m = 68.1;//kg
c = 12.5//kg/s
for i = 1:15
    v1(i) = g*m*(1 - exp(-c*s(i)/m))/c;
    v2(i) = g*m*s(i)/(c*(3.75+s(i)));
end
disp(s,"time = ")
disp(v,"measured v =")
disp(v1,"using equation(1.10) v1 = ")
disp(v2,"using equation((17.3)) v2 =  ")
plot2d(v,v1);
xtitle('v vs v1','v','v1');
```

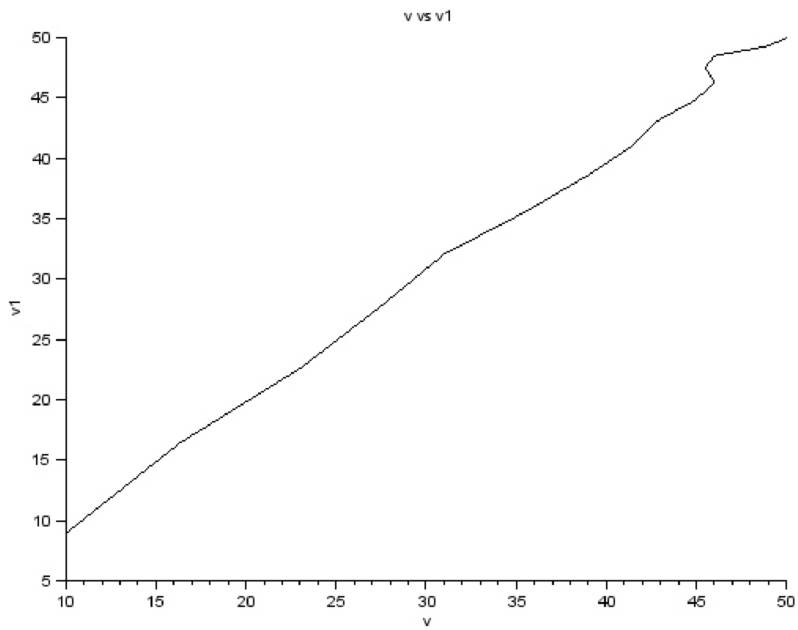

Figure 17.1: linear regression using computer

Scilab code Exa 17.3.b `linear regression using computer`

```
//clc()
s = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15];
v =
    [10,16.3,23,27.5,31,35.6,39,41.5,42.9,45,46,45.5,46,49,50];

g = 9.8//m/s^2
```

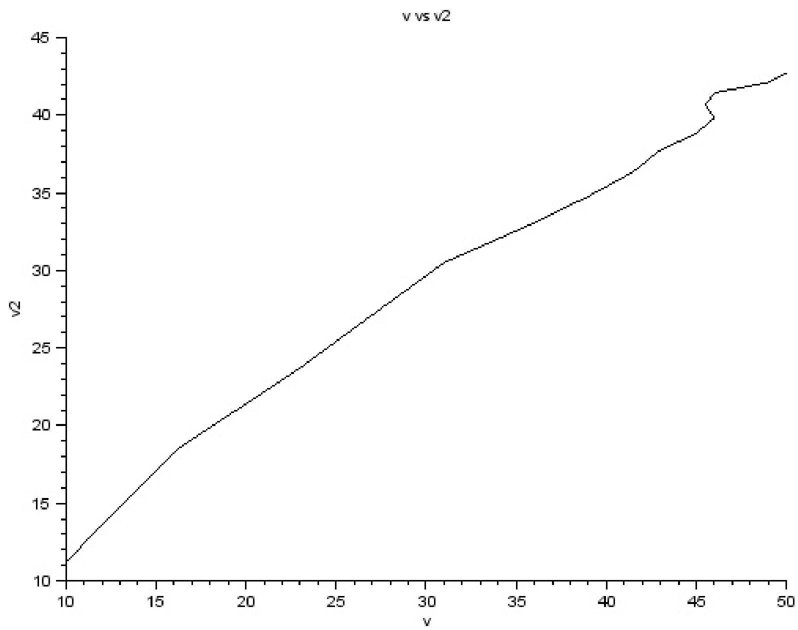

Figure 17.2: linear regression using computer

```
m = 68.1;//kg
c = 12.5//kg/s
for i = 1:15
    v1(i) = g*m*(1 - exp(-c*s(i)/m))/c;
    v2(i) = g*m*s(i)/(c*(3.75+s(i)));
end
disp(s,"time = ")
disp(v,"measured v =")
disp(v1,"using equation(1.10) v1 = ")
disp(v2,"using equation((17.3)) v2 =  ")
plot2d(v,v2);
xtitle('v vs v2','v','v2');
```

Scilab code Exa 17.4 Linearization of a power function

```
//clc()
//y = a*x^b
a1 = -0.3000;
a = 10^(a1);
b = 1.75;
disp(a)
for i=1:5
    x(i) = i;
    y(i) = a*x(i)^b;
    m(i) = log10(x(i));
    n(i) = log10(y(i));
end
disp(x(1:5),"x = ")
disp(y(1:5),"y = ")
disp(m(1:5),"m = ")
disp(n(1:5),"n = ")
plot2d(x(1:5),y(1:5))
zoom_rect([0,0,7,7])
xtitle('y vs x', 'x', 'y')
plot2d(m(1:5),n(1:5))
```

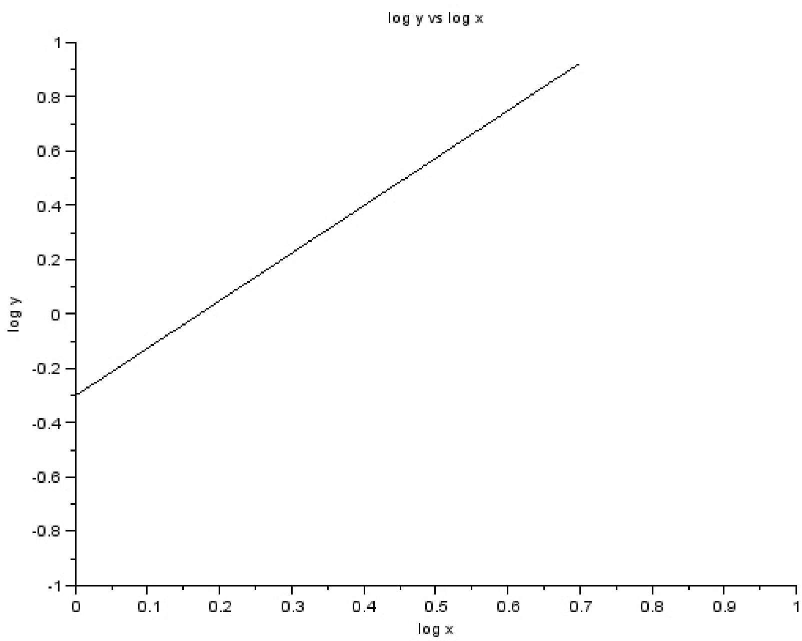

Figure 17.3: Linearization of a power function

```
zoom_rect([0,-1,1,1])
xtitle('log y vs log x', 'log x', 'log y')
```

Scilab code Exa 17.5 `polynomial regression`

```
//clc()
x = [0,1,2,3,4,5];
```

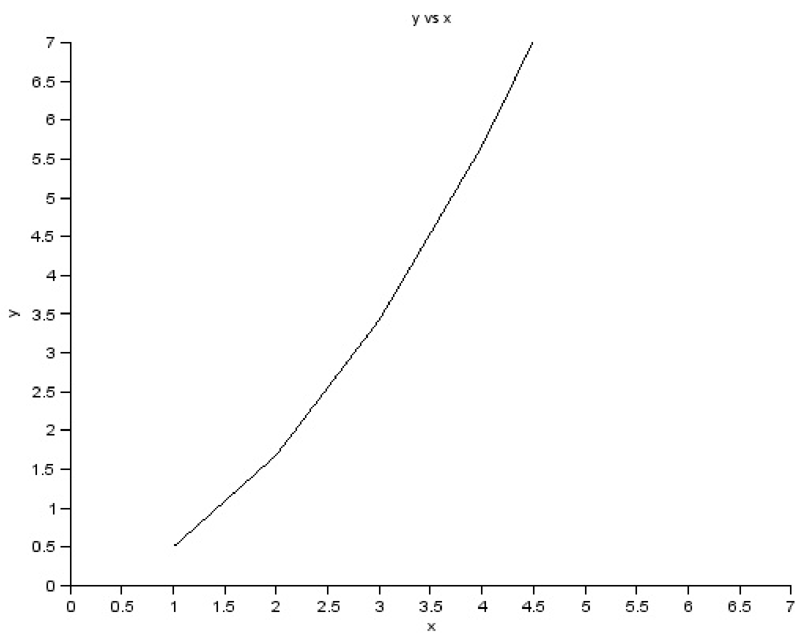

Figure 17.4: Linearization of a power function

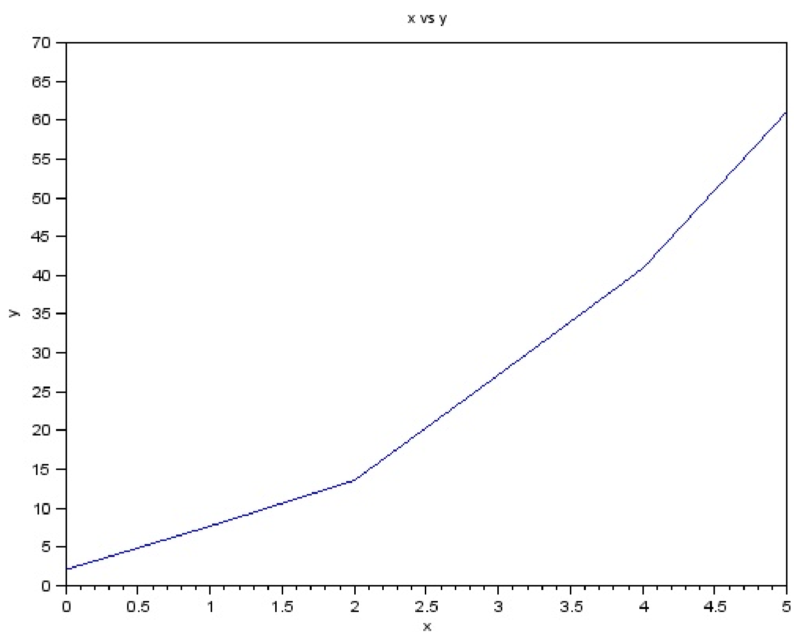

Figure 17.5: polynomial regression

```
y = [2.1,7.7,13.6,27.2,40.9,61.1];
sumy = 0;
sumx = 0;
m = 2;
n = 6;
s = 1.12;
xsqsum = 0;
xcsum = 0;
x4sum = 0;
xysum = 0;
x2ysum = 0;
rsum = 0;
usum = 0;
for i=1:6
    sumy = sumy+y(i);
    sumx = sumx+x(i);
    r(i) = (y(i) - s/n)^2;
    xsqsum = xsqsum + x(i)^2;
    xcsum = xcsum +x(i)^3;
    x4sum = x4sum + x(i)^4;
    xysum = xysum + x(i)*y(i);
    x2ysum = x2ysum + y(i)*x(i)^2;
    rsum = r(i) + rsum;
end
disp(sumy,"sum y =")
disp(sumx,"sum x =")
xavg = sumx/n;
yavg = sumy/n;
disp(xavg,"xavg = ")
disp(yavg,"yavg = ")
disp(xsqsum,"sum x^2 =")
disp(xcsum,"sum x^3 =")
disp(x4sum,"sum x^4 =")
disp(xysum,"sum x*y =")
disp(x2ysum,"sum x^2 * y =")
J = [n,sumx,xsqsum;sumx,xsqsum,xcsum;xsqsum,xcsum,x4sum];
I = [sumy;xysum;x2ysum];
```

```
X = inv(J) * I;
a0 = det(X(1,1));
a1 = det(X(2,1));
a2 = det(X(3,1));
for i=1:6
    u(i) = (y(i) - a0 - a1*x(i) - a2*x(i)^2)^2;
    usum = usum + u(i);
end
disp(r,"(yi - yavg)^2 = ")
disp(u,"(yi - a0 - a1*x - a2*x^2)^2 = ")
plot(x,y);
xtitle('x vs y','x','y');
syx = (usum/(n-3))^0,5;
disp(syx,"The standard error of the estimate based
   on regression polynomial =")
R2 = (rsum - usum)/(rsum);
disp("%",R2*100,"Percentage of original uncertainty
   that has been explained by the model = ")
```

Scilab code Exa 17.6 Multiple linear regression

```
//clc()
x1 = [0,2,2.5,1,4,7];
x2 = [0,1,2,3,6,2];
x1sum = 0;
x2sum = 0;
ysum = 0;
x12sum = 0;
x22sum = 0;
x1ysum = 0;
x2ysum = 0;
x1x2sum = 0;
n = 6;
for i=1:6
    y(i) = 5 + 4*x1(i) - 3*x2(i);
```

```
    x12(i) = x1(i)^2;
    x22(i) = x2(i)^2;
    x1x2(i) = x1(i) * x2(i);
    x1y(i) = x1(i) * y(i);
    x2y(i) = x2(i) * y(i);
    x1sum = x1sum + x1(i);
    x2sum = x2sum + x2(i);
    ysum = ysum + y(i);
    x1ysum = x1ysum + x1y(i);
    x2ysum = x2ysum + x2y(i);
    x1x2sum = x1x2sum + x1x2(i);
    x12sum = x12sum + x12(i);
    x22sum = x22sum + x22(i);
end
X = [n,x1sum,x2sum;x1sum,x12sum,x1x2sum;x2sum,
   x1x2sum,x22sum];
Y = [ysum;x1ysum;x2ysum];
Z = inv(X)*Y;
a0 = det(Z(1,1));
a1 = det(Z(2,1));
a2 = det(Z(3,1));
disp(a0,"a0 = ")
disp(a1,"a1 = ")
disp(a2,"a2 = ")
disp("Thus, y = a0 + a1*x1 + a2*x2")
```

Scilab code Exa 17.7 `Confidence interval for linear regression`

```
//clc()
//y = -0.859 + 1.032*x
Z =
   [1,10;1,16.3;1,23;1,27.5;1,31;1,35.6;1,39;1,41.5;1,42.9;1,45;1,46

for i = 1:15
    Y(i) = 9.8*68.1*(1-exp(-12.5*i/68.1))/12.5;
```

```
end
M = Z';
R = M*Z;
S = M*Y;
P = inv(R);
X = inv(R)*S;
a0 = det(X(1,1));
a1 = det(X(2,1));
disp(a0,"a0 = ")
disp(a1,"a1 = ")
sxy = 0.863403;
sa0 = (det(P(1,1)) * sxy^2)^0.5;
sa1 = (det(P(2,2)) * sxy^2)^0.5;
disp(sa0,"standard error of co efficient a0 = ")
disp(sa1,"standard error of co efficient a1 = ")
TINV = 2.160368;
a0 = [a0 - TINV*(sa0),a0 + TINV*(sa0)];
a1 = [a1 - TINV*(sa1),a1 + TINV*(sa1)];
disp(a0,"interval of a0 = ")
disp(a1,"interval of a1 = ")
```

Scilab code Exa 17.8 `Gauss Newton method`

```
//clc()
x = [0.25,0.75,1.25,1.75,2.25];
y = [0.28,0.57,0.68,0.74,0.79];
a0 = 1;
a1 = 1;
sr = 0.0248;
for i = 1:5
    pda0(i) = 1 - exp(-a1 * x(i));
    pda1(i) = a0 * x(i)*exp(-a1*x(i));
end
Z0 = [pda0(1),pda1(1);pda0(2),pda1(2);pda0(3),pda1(3);pda0(4),pda1(4);pda0(5),pda1(5)]
```

```
disp(Z0,"Z0 = ")
R = Z0'*Z0;
S = inv(R);
for i = 1:5
    y1(i) = a0 * (1-exp(-a1*x(i)));
    D(i) = y(i) - y1(i);
end
disp(D,"D = ")
M = Z0'*D;
X = S *M;
disp(X,"X = ")
a0 = a0 + det(X(1,1));
a1 = a1 + det(X(2,1));
disp(a0,"The value of a0 after 1st iteration = ")
disp(a1,"The value of a1 after 1st iteration = ")
```

Chapter 18

Interpolation

Scilab code Exa 18.1 Linear interpolation

```
//clc()
//f1(x) = f0(x) +(f(x1) - f(x0) *(x - x0)/ (x1 - x0)
x = 2;
x0 = 1;
x1 = 6;
m = 1.791759;
n = 0;
r = log(2);
f = 0 + (m - n) * (x - x0) / (x1 - x0);
disp(f,"value of ln2 for interpolation region 1 to 6
    =")
e = (r - f) * 100/r;
disp("%",e,"error by interpolation for interval[1,6]
    =")
x2 = 4;
p = 1.386294;
f = 0 + (p - n) * (x - x0) / (x2 - x0);
disp(f,"value of ln2 for interpolation region 1 to 6
    =")
e = (r - f) * 100/r;
disp("%",e,"error by interpolation for interval[1,6]
```

```
=")
```

Scilab code Exa 18.2 Quadratic interpolation

```
//clc()
x = 2;
x0 = 1;
m = 0;
x1 = 4;
n = 1.386294;
x2 = 6;
p = 1.791759;
b0 = m;
b1 = (n - m)/(x1 - x0);
b2 = ((p - n)/(x2 - x1) - (n - m)/(x1 - x0))/(x2 -
    x0);
disp(b0,"b0 = ")
disp(b1,"b1 = ")
disp(b2,"b2 = ")
f = b0 + b1*(x - x0) + b2*(x - x0)*(x - x1);
disp(f,"f(2) = ")
r = log(2);
e = (r -f)*100/r;
disp("%",e,"error = ")
```

Scilab code Exa 18.3 Newtons divided difference Interpolating polynomials

```
//clc()
x = 2;
x0 = 1;
m = 0;
x1 = 4;
n = 1.386294;
```

```
x3 = 5;
p = 1.609438;
x2 = 6;
o = 1.791759;
f01 = (m - n)/(x0 - x1);
f12 = (n - o)/(x1 - x2);
f23 = (p - o)/(x3 - x2);
f210 =  (f12 - f01)/(x2 - x0);
f321 = (f23 - f12)/(x3 - x1);
f0123 = (f321 - f210) / (x3 - x0);
b0 = m;
b1 = f01;
b2 = f210;
b3 = f0123;
disp(b0,"b0 = ")
disp(b1,"b1 = ")
disp(b2,"b2 = ")
disp(b3,"b3 = ")
f = b0 + b1*(x - x0) + b2*(x - x0)*(x - x1) + b3*(x
   - x0)*(x - x1)*(x - x2);
disp(f,"f(2) = ")
r = log(2);
e = (r -f)*100/r;
disp("%",e," error = ")
```

Scilab code Exa 18.4 Error estimation for Newtons polynomial

```
//clc()
x = 2;
x0 = 1;
m = 0;
x1 = 4;
n = 1.386294;
x3 = 5;
p = 1.609438;
```

```
x2 = 6;
o = 1.791759;
f01 = (m - n)/(x0 - x1);
f12 = (n - o)/(x1 - x2);
f23 = (p - o)/(x3 - x2);
f210 =  (f12 - f01)/(x2 - x0);
f321 = (f23 - f12)/(x3 - x1);
f0123 = (f321 - f210) / (x3 - x0);
b0 = m;
b1 = f01;
b2 = f210;
b3 = f0123;
R2 = b3 * (x - x0) * (x - x1)*(x-x2);
disp(R2,"error R2 = ")
```

Scilab code Exa 18.5 Error Estimates for Order of Interpolation

```
clc;
clear;
x=[1 4 6 5 3 1.5 2.5 3.5];
y=[0 1.3862944 1.7917595 1.6094379 1.0986123
   0.4054641 0.9162907 1.2527630];
n=8;
for i=1:n
    fdd(i,1)=y(i);
end
for j=2:n
    for i=1:n-j+1
        fdd(i,j)=(fdd(i+1,j-1)-fdd(i,j-1))/(x(i+j-1)
           -x(i));
    end
end
xterm=1;
yint(1)=fdd(1,1);

```

```
for order=2:n
    xterm=xterm*(2-x(order-1));
    yint2=yint(order-1)+fdd(1,order)*xterm;
    Ea(order-1)=yint2-yint(order-1);
    yint(order)=yint2;

end
disp(yint,"F(x)=")
disp(Ea,"Ea=")
```

Scilab code Exa 18.6 Lagrange interpolating polynomials

```
//clc()
x = 2;
x0 = 1;
m = 0;
x1 = 4;
n = 1.386294;
x2 = 6;
p = 1.791759;
f1 = (x - x1)*m/((x0 - x)) + (x- x0) * n/(x1 - x0);
disp(f1,"first order polynomial f1(2) = ")
f2 = (x - x1)*(x - x2)*m/((x0 - x1)*(x0 - x2)) + (x
    - x0)*(x - x2)*n/((x1-x0)*(x1-x2)) + (x - x0)*(x
    - x1)*p/((x2 - x0)*(x2 - x1));
disp(f2,"second order polynomial f2(2) = ")
```

Scilab code Exa 18.7 Lagrange interpolation using computer

```
//clc()
z = 10;
x =[1,3,5,7,13];
v = [800,2310,3090,3940,4755];
```

```
f1 = (z - x(5)) * v(4) / (x(4) - x(5)) + (z - x(4)) * v(5) / (x(5) - x(4));
f2 = (z - x(4))*(z - x(5)) *v(3)/((x(3) - x(4))*(x(3) - x(5)))+(z - x(3))*(z - x(5)) *v(4)/((x(4) - x(3))*(x(4) - x(5)))+(z - x(4))*(z - x(3)) *v(5)/((x(5) - x(4))*(x(5) - x(3)));
f3 = (z - x(3))*(z - x(4))*(z - x(5)) *v(2)/((x(2) - x(4))*(x(2) - x(5))*(x(2) - x(3)))+(z - x(4))*(z - x(2))*(z - x(5)) *v(3)/((x(3) - x(2))*(x(3) - x(5))*(x(3) - x(4)))+(z - x(2))*(z - x(3))*(z - x(5)) *v(4)/((x(4) - x(3))*(x(4) - x(2))*(x(4) - x(5)))+ (z - x(3))*(z - x(4))*(z - x(2)) *v(5)/((x(5) - x(4))*(x(5) - x(2))*(x(5) - x(3)));
f4 = (z - x(2))*(z - x(3))*(z - x(4))*(z - x(5)) *v(1)/((x(1) - x(2))*(x(1) - x(4))*(x(1) - x(5))*(x(1) - x(3)))+ (z - x(1))*(z - x(3))*(z - x(4))*(z - x(5)) *v(2)/((x(2) - x(1))*(x(2) - x(4))*(x(2) - x(5))*(x(2) - x(3)))+(z - x(1))*(z - x(4))*(z - x(2))*(z - x(5)) *v(3)/((x(3) - x(1))*(x(3) - x(2))*(x(3) - x(5))*(x(3) - x(4)))+(z - x(1))*(z - x(2))*(z - x(3))*(z - x(5)) *v(4)/((x(4) - x(1))*(x(4) - x(3))*(x(4) - x(2))*(x(4) - x(5)))+ (z - x(1))*(z - x(3))*(z - x(4))*(z - x(2)) *v(5)/((x(5) - x(1))*(x(5) - x(4))*(x(5) - x(2))*(x(5) - x(3)));
disp(f1,"Velocity at 10 sec by first order interpolation = ")
disp(f2,"Velocity at 10 sec by second order interpolation = ")
disp(f3,"Velocity at 10 sec by third order interpolation = ")
disp(f4,"Velocity at 10 sec by fourth order interpolation = ")
```

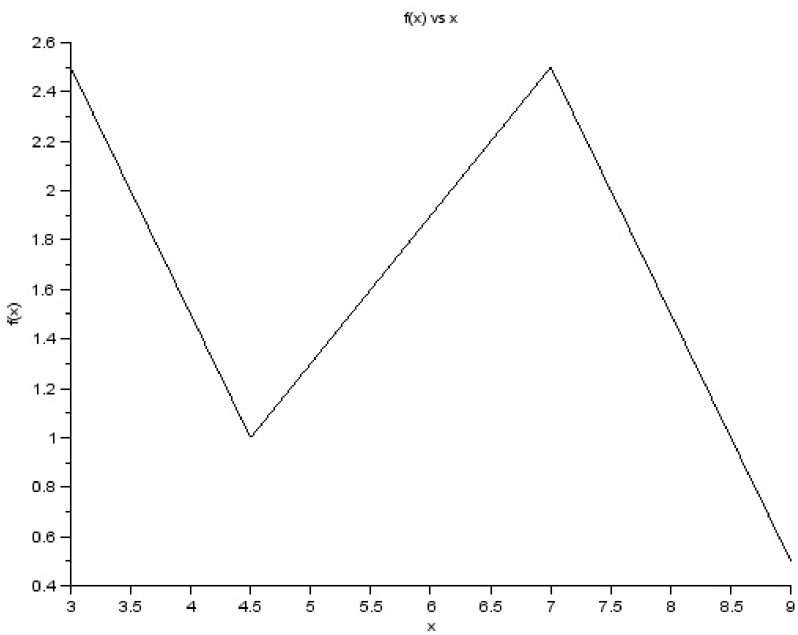

Figure 18.1: First order splines

Scilab code Exa 18.8 First order splines

```
//clc()
x = [3,4.5,7,9];
fx = [2.5,1,2.5,0.5];
m1 = (fx(2) - fx(1))/(x(2) - x(1));
m2 = (fx(3) - fx(2))/(x(3) - x(2));
m3 = (fx(4) - fx(3))/(x(4) - x(3));
x1 = [3,4.5];
x2 = [4.5,7];
x3 = [7,9];
plot2d(x1,m1*x1+5.5);
plot2d(x2,m2*x2-1.7);
plot2d(x3,m3*x3+9.5);
xtitle("f(x) vs x","x","f(x)")
r = 5
z = m2*r -1.7;
disp(z,"The value at x = 5 is")
```

Scilab code Exa 18.9 Quadratic splines

```
//clc()
x = [3,4.5,7,9];
fx = [2.5,1,2.5,0.5];
p = 4;
n = 3;
uk = n*(p-1);
c = 2*n - 2;
//20.25*a1 + 4.5*b1 + c1 = 1
//20.25*a2 + 4.5*b2 + c2 = 1
//49*a2 + 7*b2 + c2 = 2.5
//49*a3 + 7*b3 + c3 = 2.5
//9a1 + 3b1 + c1 = 2.5
```

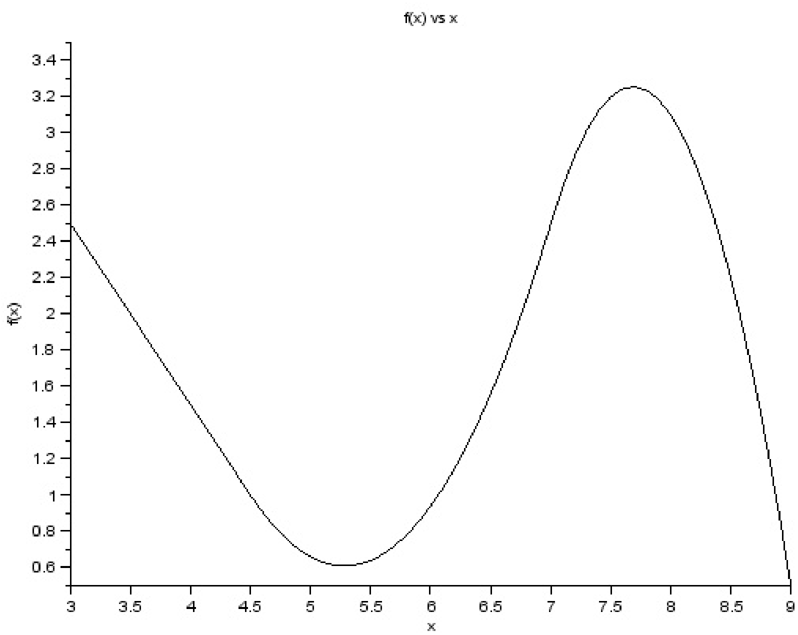

Figure 18.2: Quadratic splines

```
//81*a3 + 9*b3 + c3 = 0.5
//9*a1 + b1 = 9*a2 + b2
//14a2 + b2 = 14 *a3 + b3
a1 = 0;
//thus we have above 9 equations and 9 unknowns a1,a2,a3,b1,b2,b3,c1,c2,c3
//thus we get
A = [20.25,4.5,1,0,0,0,0,0,0;0,0,0,20.25,4.5,1,0,0,0;0,0,0,49,7,1,0,0,
disp(A,"A = ")
B = [1;1;2.5;2.5;2.5;0.5;0;0;0];
disp(B,"B =")
X = inv(A)*B;
a1 = det(X(1,1));
b1 = det(X(2,1));
c1 = det(X(3,1));
a2 = det(X(4,1));
b2 = det(X(5,1));
c2 = det(X(6,1));
a3 = det(X(7,1));
b3 = det(X(8,1));
c3 = det(X(9,1));
disp(a1,"a1 = ")
disp(b1,"b1 = ")
disp(c1,"c1 = ")
disp(a2,"a2 = ")
disp(b2,"b2 = ")
disp(c2,"c2 = ")
disp(a3,"a3 = ")
disp(b3,"b3 = ")
disp(c3,"c3 = ")
//thus,f1(x) = -x + 5.5                        3 < x < 4.5
//f2(x) = 0.64*x^2 -6.76*x + 18.46              4.5 < x < 7
//f3(x) = -1.6*x^2 + 24.6*x - 91.3              7 < x < 9
```

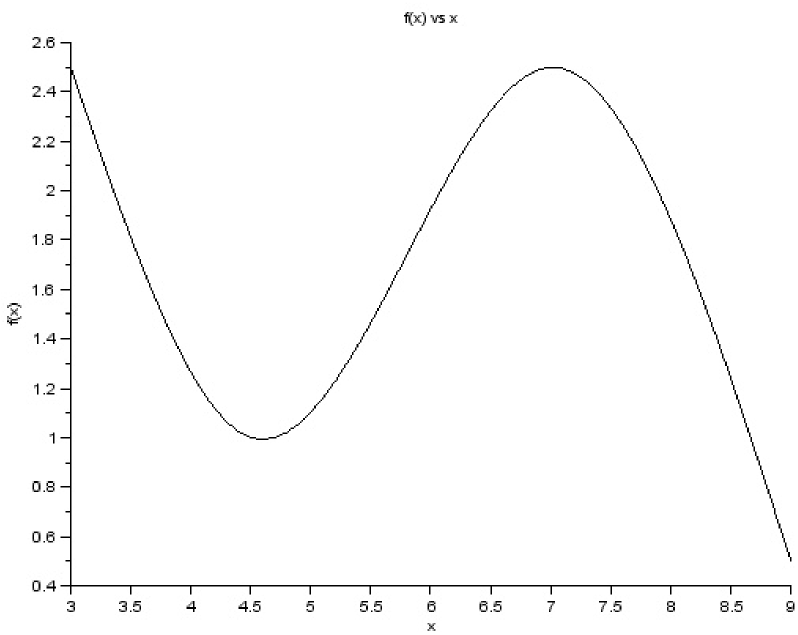

Figure 18.3: Cubic splines

```
x1 = 3:0.1:4.5;
x2 = 4.5:0.1:7;
x3 = 7:0.1:9;
plot2d(x1,-x1 + 5.5);
plot2d(x2,0.64*x2^2 -6.76*x2+ 18.46);
plot2d(x3,-1.6*x3^2 + 24.6*x3 - 91.3);
xtitle("f(x) vs x","x","f(x)")
x = 5;
fx = 0.64*x^2 -6.76*x + 18.46;
disp(fx,"The value at x = 5 is")
```

Scilab code Exa 18.10 Cubic splines

```
//clc()
x = [3,4.5,7,9];
fx = [2.5,1,2.5,0.5];
//we get the following equations for cubic splines
//8*f"(4.5) + 2.5*f"(7) = 9.6
//2.5*f"(4.5) + 9*f"(7) = -9.6
//above two equations give
m = 1.67909;//(m = f"(4.5))
n = -1.53308;//(n = f"(7))
//this values can be used to yield the final
    equation
//f1(x) = 0.186566 * (x - 3)^3 + 1.66667*(4.5 - x) +
    0.246894*(x - 3)
//in similar manner other equations can be found too
//f2(x) = 0.111939(7 - x)^3 - 0.102205*(x - 4.5)^3 -
    0.299621 * (7 - x) + 1.638783 * (x - 4.5)
//f3(x) = -0.127757*(9 - x)^3 + 1.761027 *(9 - x) +
    0.25*(x - 7)
x1 = 3:0.1:4.5;
x2 = 4.5:0.1:7;
x3 = 7:0.1:9;
plot2d(x1,0.186566 * (x1 - 3)^3 + 1.66667*(4.5 - x1)
    + 0.246894*(x1 - 3));
plot2d(x2,0.111939*(7 - x2)^3 - 0.102205*(x2 - 4.5)
    ^3 - 0.299621 * (7 - x2) + 1.638783 * (x2 - 4.5))
    ;
plot2d(x3,-0.127757*(9 - x3)^3 + 1.761027 *(9 - x3)
    + 0.25*(x3 - 7));
xtitle("f(x) vs x","x","f(x)")
x = 5;
fx = 0.111939*(7 - x)^3 - 0.102205*(x - 4.5)^3 -
    0.299621 * (7 - x) + 1.638783 * (x - 4.5);
disp(fx,"The value at x = 5 is")
```

Chapter 19

Fourier Approximation

Scilab code Exa 19.1 `Least Square Fit`

```
clc;
clear;
function y=f(t)
    y=1.7+cos(4.189*t+1.0472)
endfunction
deltat=0.15;
t1=0;
t2=1.35;
omega=4.189;
del=(t2-t1)/9;
for i=1:10
    t(i)=t1+del*(i-1);
end
sumy=0;
suma=0;
sumb=0;
for i=1:10
    y(i)=f(t(i));
    a(i)=y(i)*cos(omega*t(i));
    b(i)=y(i)*sin(omega*t(i));
    sumy=sumy+y(i);
```

```
    suma=suma+a(i);
    sumb=sumb+b(i);
end
A0=sumy/10;
A1=2*suma/10;
B1=2*sumb/10;
disp("The least square fit is y=A0+A1*cos(w0*t)+A2*
    sin(w0*t), where")
disp(A0,"A0=")
disp(A1,"A1=")
disp(B1,"B1=")
theta=atan(-B1/A1);
C1=(A1^2 + B1^2)^0.5;
disp("Alternatively, the least square fit can be
    expressed as")
disp("y=A0+C1*cos(w0*t + theta), where")
disp(A0,"A0=")
disp(theta,"Theta=")
disp(C1,"C1=")
disp("Or")
disp("y=A0+C1*sin(w0*t + theta + pi/2), where")
disp(A0,"A0=")
disp(theta,"Theta=")
disp(C1,"C1=")
```

Scilab code Exa 19.2 Continuous Fourier Series Approximation

```
clc;
clear;
a0=0;
//f(t)=-1 for -T/2 to -T/4
//f(t)=1 for -T/4 to T/4
//f(t)=-1 for T/4 to T/2
//ak=2/T* (integration of f(t)*cos(w0*t) from -T/2
    to T/2)
```

```
//ak=2/T*((integration of f(t)*cos(w0*t) from -T/2
   to -T/4) + (integration of f(t)*cos(w0*t) from -T
   /4 to T/4) + (integration of f(t)*cos(w0*t) from
   T/4 to T/2))
//Therefore,
//ak=4/(k*%pi) for k=1,5,9,.....
//ak=-4/(k*%pi) for k=3,7,11,.....
//ak=0 for k=even integers
//similarly we find the b's.
//all the b's=0
disp("The fourier approximtion is:")
disp("4/(%pi)*cos(w)*t) - 4/(3*%pi)*cos(3*(w)*t) +
   4/(5*%pi)*cos(5*(w)*t) - 4/(7*%pi)*cos(7*(w)*t) +
    .....")
```

Scilab code Exa 19.3 Trendline

```
clc;
clear;
x=0.5:0.5:5.5;
for i=1:11
    y(i)=0.9846*log(x(i))+1.0004;
end
plot(x,y)
xtitle("y vs x","x","y")
```

Scilab code Exa 19.4 Data Analysis

```
clc;
clear;
s=[0.0002 0.0002 0.0005 0.0005 0.001 0.001];
```

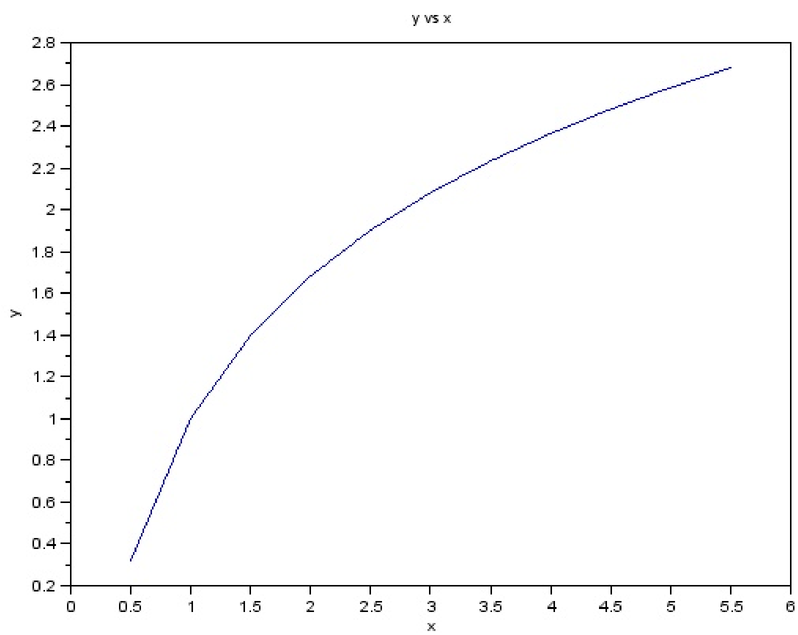

Figure 19.1: Trendline

```
r=[0.2 0.5 0.2 0.5 0.2 0.5];
u=[0.25; 0.5; 0.4; 0.75; 0.5; 1];
logs=log10(s);
logr=log10(r);
logu=log10(u);
for i=1:6
    m(i,1)=1;
    m(i,2)=logs(i);
    m(i,3)=logr(i);
end

a=m\logu;
disp(10^a(1),"alpha=")
disp(a(2),"sigma=")
disp(a(3),"rho=")
```

Scilab code Exa 19.5 Curve Fitting

```
clc;
clear;
x=0:10;
y=sin(x);
xi=0:.25:10;
//part a
yi=interp1(x,y,xi);
plot2d(xi,yi)
xtitle("y vs x (part a)","x","y")
//part b
//fitting x and y in a fifth order polynomial
clf();
p=[0.0008 -0.0290 0.3542 -1.6854 2.586 -0.0915];

for i=1:41
    yi(i)=p(1)*(xi(i)^5)+p(2)*(xi(i)^4)+p(3)*(xi(i)
       ^3)+p(4)*(xi(i)^2)+p(5)*(xi(i))+p(6);
```

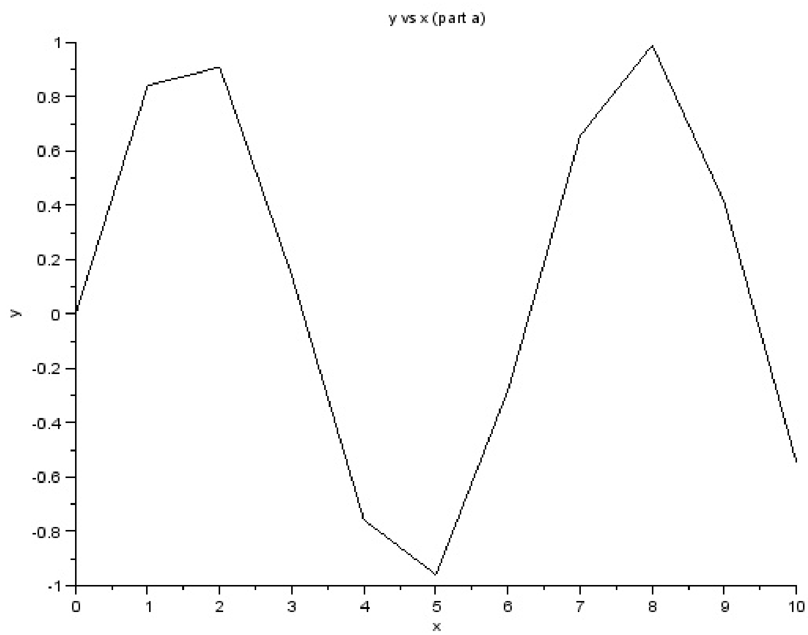

Figure 19.2: Curve Fitting

```
end
plot2d(xi,yi);
xtitle("y vs x (part b)","x","y")
//part c
clf();
d=splin(x,y)
yi-interp(xi,x,y,d)
plot2d(xi,yi)
xtitle("y vs x (part c)","x","y")
```

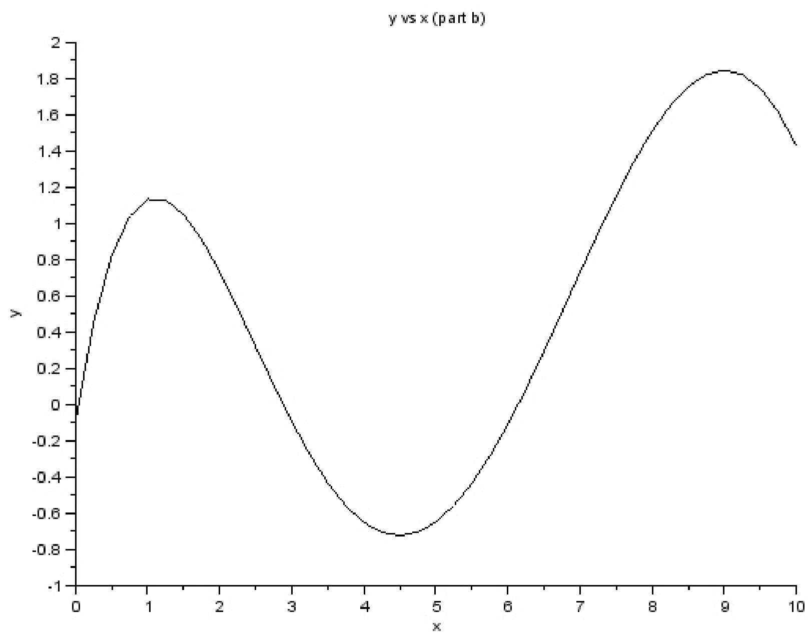

Figure 19.3: Curve Fitting

Scilab code Exa 19.6 Polynomial Regression

```
clc;
clear;
x=[0.05 0.12 0.15 0.3 0.45 0.7 0.84 1.05];
y=[0.957 0.851 0.832 0.72 0.583 0.378 0.295 0.156];
sx=sum(x);
sxx=sum(x.*x);
sx3=sum(x.*x.*x);
sx4=sum(x.*x.*x.*x);
sx5=sum(x.*x.*x.*x.*x);
sx6=sum(x.*x.*x.*x.*x.*x);
n=8;
sy=sum(y);
sxy=sum(x.*y);
sx2y=sum(x.*x.*y);
sx3y=sum(x.*x.*x.*y);
m=[n sx sxx sx3;sx sxx sx3 sx4;sxx sx3 sx4 sx5;sx3
    sx4 sx5 sx6];
p=[sy;sxy;sx2y;sx3y];
a=m\p;
disp("The cubic polynomial is y=a0 + a1*x + a2*x^2 +
    a3*x^3, where a0, a1, a2 and a3 are")
disp(a)
```

Chapter 21

Newton Cotes Integration Formulas

Scilab code Exa 21.1 Single trapezoidal rule

```
clc;
clear;
function y=f(x)
    y=(0.2+25*x-200*x^2+675*x^3-900*x^4+400*x^5)
endfunction
tval=1.640533;
a=0;
b=0.8;
fa=f(a);
fb=f(b);
l=(b-a)*((fa+fb)/2);
Et=tval-l;//error
et=Et*100/tval;//percent relative error

//by using approximate error estimate

//the second derivative of f
function y=g(x)
    y=-400+4050*x-10800*x^2+8000*x^3
```

```
endfunction
f2x=intg(0,0.8,g)/(b-a);//average value of second derivative
Ea=-(1/12)*(f2x)*(b-a)^3;
disp(Et,"The Error Et=")
disp("%",et,"The percent relative error et=")
disp(Ea,"The approximate error estimate without using the true value=")
```

Scilab code Exa 21.2 `Multiple trapezoidal rule`

```
clc;
clear;
function y=f(x)
    y=(0.2+25*x-200*x^2+675*x^3-900*x^4+400*x^5)
endfunction
a=0;
b=0.8;
tval=1.640533;
n=2;
h=(b-a)/n;
fa=f(a);
fb=f(b);
fh=f(h);
l=(b-a)*(fa+2*fh+fb)/(2*n);
Et=tval-l;//error
et=Et*100/tval;//percent relative error

//by using approximate error estimate

//the second derivative of f
function y=g(x)
    y=-400+4050*x-10800*x^2+8000*x^3
endfunction
f2x=intg(0,0.8,g)/(b-a);//average value of second
```

```
   derivative
Ea=-(1/12)*(f2x)*(b-a)^3/(n^2);
disp(Et,"The Error Et=")
disp("%",et,"The percent relative error et=")
disp(Ea,"The approximate error estimate without
   using the true value=")
```

Scilab code Exa 21.3 Evaluating Integrals

```
clc;
clear;
g=9.8;//m/s^2; acceleration due to gravity
m=68.1;//kg
c=12.5;//kg/sec; drag coefficient
function v=f(t)
    v=g*m*(1-exp(-c*t/m))/c
endfunction
tval=289.43515;//m
a=0;
b=10;
fa=f(a);
fb=f(b);
for i=10:10:20
    n=i;
    h=(b-a)/n;
    disp(i,"No. of segments=")
    disp(h,"Segment size=")
    j=a+h;
    s=0;
    while j<b
        s=s+f(j);
        j=j+h;
    end
    l=(b-a)*(fa+2*s+fb)/(2*n);
    Et=tval-l;//error
```

```
    et=Et*100/tval;//percent relative error
    disp("m",l,"Estimated d=")
    disp(et,"et(%)")
    disp("
      ————————————————————————————————————————————————————————————
      ")
end
for i=50:50:100
    n=i;
    h=(b-a)/n;
    disp(i,"No. of segments=")
    disp(h,"Segment size=")
    j=a+h;
    s=0;
    while j<b
        s=s+f(j);
        j=j+h;
    end
    l=(b-a)*(fa+2*s+fb)/(2*n);
    Et=tval-l;//error
    et=Et*100/tval;//percent relative error
    disp("m",l,"Estimated d=")
    disp(et,"et(%)")
    disp("
      ————————————————————————————————————————————————————————————
      ")
end
for i=100:100:200
    n=i;
    h=(b-a)/n;
    disp(i,"No. of segments=")
    disp(h,"Segment size=")
    j=a+h;
    s=0;
    while j<b
        s=s+f(j);
        j=j+h;
    end
```

```
    l=(b-a)*(fa+2*s+fb)/(2*n);
    Et=tval-l;//error
    et=Et*100/tval;//percent relative error
    disp("m",l,"Estimated d=")
    disp(et,"et(%)")
    disp("------------------------------------------------------------")
end
for i=200:300:500
    n=i;
    h=(b-a)/n;
    disp(i,"No. of segments=")
    disp(h,"Segment size=")
    j=a+h;
    s=0;
    while j<b
        s=s+f(j);
        j=j+h;
    end
    l=(b-a)*(fa+2*s+fb)/(2*n);
    Et=tval-l;//error
    et=Et*100/tval;//percent relative error
    disp("m",l,"Estimated d=")
    disp(et,"et(%)")
    disp("------------------------------------------------------------")
end
for i=1000:1000:2000
    n=i;
    h=(b-a)/n;
    disp(i,"No. of segments=")
    disp(h,"Segment size=")
    j=a+h;
    s=0;
    while j<b
        s=s+f(j);
```

```
        j=j+h;
    end
    l=(b-a)*(fa+2*s+fb)/(2*n);
    Et=tval-l;//error
    et=Et*100/tval;//percent relative error
    disp("m",l,"Estimated d=")
    disp(et,"et(%)")
    disp("
    ----------------------------------------------------------
    ")
end
for i=2000:3000:5000
    n=i;
    h=(b-a)/n;
    disp(i,"No. of segments=")
    disp(h,"Segment size=")
    j=a+h;
    s=0;
    while j<b
        s=s+f(j);
        j=j+h;
    end
    l=(b-a)*(fa+2*s+fb)/(2*n);
    Et=tval-l;//error
    et=Et*100/tval;//percent relative error
    disp("m",l,"Estimated d=")
    disp(et,"et(%)")
    disp("
    ----------------------------------------------------------
    ")
end
for i=5000:5000:10000
    n=i;
    h=(b-a)/n;
    disp(i,"No. of segments=")
    disp(h,"Segment size=")
    j=a+h;
    s=0;
```

```
    while j<b
        s=s+f(j);
        j=j+h;
    end
    l=(b-a)*(fa+2*s+fb)/(2*n);
    Et=tval-l;//error
    et=Et*100/tval;//percent relative error
    disp("m",l,"Estimated d=")
    disp(et,"et(%)")
    disp("
        ____________________________________________________________________
        ")
end
```

Scilab code Exa 21.4 Single Simpsons 1 by 3 rule

```
clc;
clear;
function y=f(x)
    y=(0.2+25*x-200*x^2+675*x^3-900*x^4+400*x^5)
endfunction
a=0;
b=0.8;
tval=1.640533;
n=2;
h=(b-a)/n;
fa=f(a);
fb=f(b);
fh=f(h);
l=(b-a)*(fa+4*fh+fb)/(3*n);
disp(l,"l=")
Et=tval-l;//error
et=Et*100/tval;//percent relative error

//by using approximate error estimate
```

```
//the fourth derivative of f
function y=g(x)
    y=-21600+48000*x
endfunction
f4x=intg(0,0.8,g)/(b-a);//average value of fourth
    derivative
Ea=-(1/2880)*(f4x)*(b-a)^5;
disp(Et,"The Error Et=")
disp("%",et,"The percent relative error et=")
disp(Ea,"The approximate error estimate without
    using the true value=")
```

Scilab code Exa 21.5 Multiple Simpsons 1 by 3 rule

```
clc;
clear;
function y=f(x)
    y=(0.2+25*x-200*x^2+675*x^3-900*x^4+400*x^5)
endfunction
a=0;
b=0.8;
tval=1.640533;
n=4;
h=(b-a)/n;
fa=f(a);
fb=f(b);
j=a+h;
s=0;
count=1;
while j<b
    if (-1)^count==-1 then
        s=s+4*f(j);
    else
        s=s+2*f(j);
```

```
    end
    count=count+1;
    j=j+h;
end
l=(b-a)*(fa+s+fb)/(3*n);
disp(l,"l=")
Et=tval-l;//error
et=Et*100/tval;//percent relative error

//by using approximate error estimate

//the fourth derivative of f
function y=g(x)
    y=-21600+48000*x
endfunction
f4x=intg(0,0.8,g)/(b-a);//average value of fourth
   derivative
Ea=-(1/(180*4^4))*(f4x)*(b-a)^5;
disp(Et,"The Error Et=")
disp("%",et,"The percent relative error et=")
disp(Ea,"The approximate error estimate without
   using the true value=")
```

Scilab code Exa 21.6 Simpsons 3 by 8 rule

```
clc;
clear;
function y=f(x)
    y=(0.2+25*x-200*x^2+675*x^3-900*x^4+400*x^5)
endfunction
a=0;
b=0.8;
tval=1.640533;
//part a
n=3;
```

```
h=(b-a)/n;
fa=f(a);
fb=f(b);
j=a+h;
s=0;
count=1;
while j<b
    s=s+3*f(j);
    count=count+1;
    j=j+h;
end
l=(b-a)*(fa+s+fb)/(8);
disp("Part A:")
disp(l,"l=")
Et=tval-l;//error
et=Et*100/tval;//percent relative error

//by using approximate error estimate

//the fourth derivative of f
function y=g(x)
    y=-21600+48000*x
endfunction
f4x=intg(0,0.8,g)/(b-a);//average value of fourth derivative
Ea=-(1/6480)*(f4x)*(b-a)^5;
disp(Et,"The Error Et=")
disp("%",et,"The percent relative error et=")
disp(Ea,"The approximate error estimate without using the true value=")

//part b
n=5;
h=(b-a)/n;
l1=(a+2*h-a)*(fa+4*f(a+h)+f(a+2*h))/6;
l2=(a+5*h-a-2*h)*(f(a+2*h)+3*(f(a+3*h)+f(a+4*h))+fb)/8;
l=l1+l2;
```

```
disp("
    ----------------------------------------------------------------------
    ")
disp("Part B:")
disp(l,"l=")
Et=tval-l;//error
et=Et*100/tval;//percent relative error
disp(Et,"The Error Et=")
disp("%",et,"The percent relative error et=")
```

Scilab code Exa 21.7 `Unequal Trapezoidal segments`

```
clc;
clear;
function y=f(x)
    y=(0.2+25*x-200*x^2+675*x^3-900*x^4+400*x^5)
endfunction
tval=1.640533;
x=[0 0.12 0.22 0.32 0.36 0.4 0.44 0.54 0.64 0.7
    0.8];
for i=1:11
    func(i)=f(x(i));
end
l=0;
for i=1:10
    l=l+(x(i+1)-x(i))*(func(i)+func(i+1))/2;
end
disp(l,"l=")
Et=tval-l;//error
et=Et*100/tval;//percent relative error
disp(Et,"The Error Et=")
disp("%",et,"The percent relative error et=")
```

Scilab code Exa 21.8 Simpsons Uneven data

```
clc;
clear;
function y=f(x)
    y=(0.2+25*x-200*x^2+675*x^3-900*x^4+400*x^5)
endfunction
tval=1.640533;
x=[0 0.12 0.22 0.32 0.36 0.4 0.44 0.54 0.64 0.7 0.8];
for i=1:11
    func(i)=f(x(i));
end
l1=(x(2)-x(1))*((f(x(1))+f(x(2)))/2);
l2=(x(4)-x(2))*(f(x(4))+4*f(x(3))+f(x(2)))/6;
l3=(x(7)-x(4))*(f(x(4))+3*(f(x(5))+f(x(6)))+f(x(7)))/8;
l4=(x(9)-x(7))*(f(x(7))+4*f(x(8))+f(x(9)))/6;
l5=(x(10)-x(9))*((f(x(10))+f(x(9)))/2);
l6=(x(11)-x(10))*((f(x(11))+f(x(10)))/2);
l=l1+l2+l3+l4+l5+l6;
disp(l,"l=")
Et=tval-l;//error
et=Et*100/tval;//percent relative error
disp(Et,"The Error Et=")
disp("%",et,"The percent relative error et=")
```

Scilab code Exa 21.9 Average Temperature Determination

```
clc;
clear;
function t=f(x,y)
    t=2*x*y+2*x-x^2-2*y^2+72
endfunction
len=8;//m, length
```

```
wid=6; //m, width
a=0;
b=len;
n=2;
h=(b-a)/n;
a1=0;
b1=wid;
h1=(b1-a1)/n;

fa=f(a,0);
fb=f(b,0);
fh=f(h,0);
lx1=(b-a)*(fa+2*fh+fb)/(2*n);

fa=f(a,h1);
fb=f(b,h1);
fh=f(h,h1);
lx2=(b-a)*(fa+2*fh+fb)/(2*n);

fa=f(a,b1);
fb=f(b,b1);
fh=f(h,b1);
lx3=(b-a)*(fa+2*fh+fb)/(2*n);

l=(b1-a1)*(lx1+2*lx2+lx3)/(2*n);

avg_temp=l/(len*wid);
disp(avg_temp,"The average termperature is=")
```

Chapter 22

Integration of equations

Scilab code Exa 22.1 Error corrections of the trapezoidal rule

```
//clc()
h = [0.8,0.4,0.2];
I = [0.1728,1.0688,1.4848];
E = [89.5,34.9,9.5];
I1 = 4 * I(2) / 3 - I(1) / 3;
t = 1.640533;
et1 = t - I1 ;
Et1 = et1 * 100/t;
disp("%",Et1,"Error of the improved integral for
   segment 1 and 2 = ")
I2 = 4 * I(3) / 3 - I(2) / 3;
et2 = t - I2 ;
Et2 = et2 * 100/t;
disp("%",Et2,"Error of the improved integral for
   segment 4 and 2 = ")
```

Scilab code Exa 22.2 Higher order error correction of integral estimates

```
//clc()
I1 = 1.367467;
I2 = 1.623467;
I = 16 * I2 /15 - I1 / 15;
disp(I,"Obtained integral which is the correct
   answer till the seventh decimal")
```

Scilab code Exa 22.3 `Two point gauss legendre formulae`

```
//clc()
//f(x) = 0.2 + 25*x - 200*x^2 + 675*x^3 - 900*x^4 +
   400*x^5
// for using two point gauss legendre formulae, the
   intervals have to be changed to -1 and 1
//therefore, x = 0.4 + 0.4 * xd
//thus the integral is transferred to
//(0.2 + 25*(0.4+0.4*x) - 200*(0.4 + 0.4*x)^2 +
   675*(0.4 + 0.4*x)^3 - 900*(0.4 + 0.4*x)^4 +
   400*(0.4 + 0.4*x)^5)*0.4
//for three point gauss legendre formulae
x1 = -(1/3) ^ 0.5;
x2 = (1/3) ^ 0.5;
I1 = (0.2 + 25*(0.4+0.4*x1) - 200*(0.4 + 0.4*x1)^2 +
    675*(0.4 + 0.4*x1)^3 - 900*(0.4 + 0.4*x1)^4 +
   400*(0.4 + 0.4*x1)^5)*0.4;
I2 = (0.2 + 25*(0.4+0.4*x2) - 200*(0.4 + 0.4*x2)^2 +
    675*(0.4 + 0.4*x2)^3 - 900*(0.4 + 0.4*x2)^4 +
   400*(0.4 + 0.4*x2)^5)*0.4;
I = I1 + I2;
disp(I,"Integral obtained using gauss legendre
   formulae =")
t = 1.640533;
e = (t - I)*100/t;
disp("%",e,"error = ")
```

Scilab code Exa 22.4 Three point gauss legendre method

```
//clc()
//f(x) = 0.2 + 25*x - 200*x^2 + 675*x^3 - 900*x^4 +
   400*x^5
// for using three point gauss legendre formulae,
   the intervals have to be changed to -1 and 1
//therefore, x = 0.4 + 0.4 * xd
//thus the integral is transferred to
//(0.2 + 25*(0.4+0.4*x) - 200*(0.4 + 0.4*x)^2 +
   675*(0.4 + 0.4*x)^3 - 900*(0.4 + 0.4*x)^4 +
   400*(0.4 + 0.4*x)^5)*0.4
//for three point gauss legendre formulae
x1 = -0.7745967;
x2 = 0;
x3 = 0.7745967;
c0 = 0.5555556;
c1 = 0.8888889;
c2 = 0.5555556;
I1 = (0.2 + 25*(0.4+0.4*x1) - 200*(0.4 + 0.4*x1)^2 +
    675*(0.4 + 0.4*x1)^3 - 900*(0.4 + 0.4*x1)^4 +
   400*(0.4 + 0.4*x1)^5)*0.4;
I2 = (0.2 + 25*(0.4+0.4*x2) - 200*(0.4 + 0.4*x2)^2 +
    675*(0.4 + 0.4*x2)^3 - 900*(0.4 + 0.4*x2)^4 +
   400*(0.4 + 0.4*x2)^5)*0.4;
I3 = (0.2 + 25*(0.4+0.4*x3) - 200*(0.4 + 0.4*x3)^2 +
    675*(0.4 + 0.4*x3)^3 - 900*(0.4 + 0.4*x3)^4 +
   400*(0.4 + 0.4*x3)^5)*0.4;
I = c0 * I1 + c1 * I2 + c2 * I3;
disp(I,"integral obtained using three point gauss
   legendre formulae = ")
```

Scilab code Exa 22.5 Applying Gauss Quadrature to the falling Parachutist problem

```
//clc()
//f(t) = g*m*(int(0,10,(1-exp(-c*t/m))))/c
//for using gauss quadrature method, limits are
   changed to -1 to 1 by replcing x = 5 + 5*xd
//the new integral obtained is
//(1 - exp(-c*(5 + 5*x)/m ))*5
g = 9.8;
c = 12.5;
m = 68.1;
//for two point method
x1 = -(1/3)^0.5;
x2 = (1/3)^0.5;
I1 = g*m*(1 - exp(-c*(5 + 5*x1)/m ))*5 / c;
I2 = g*m*(1 - exp(-c*(5 + 5*x2)/m ))*5 / c;
I = I1 + I2;
disp(I,"integral by two point method = ")
x1 = -0.7745967;
x2 = 0;
x3 = 0.7745967;
c0 = 0.5555556;
c1 = 0.8888889;
c2 = 0.5555556;
I1 = g*m*(1 - exp(-c*(5 + 5*x1)/m ))*5 / c;
I2 = g*m*(1 - exp(-c*(5 + 5*x2)/m ))*5 / c;
I3 = g*m*(1 - exp(-c*(5 + 5*x3)/m ))*5 / c;
I = c0*I1 + c1 * I2 + c2 * I3;
disp(I,"integral by three point method =")
x1 = -0.861136312;
x2 = -0.339981044;
x3 = 0.339981044;
x4 = 0.861136312;
c1 = 0.3478548;
c2 = 0.6521452;
c3 = 0.6521452;
c4 = 0.3478548;
I1 = g*m*(1 - exp(-c*(5 + 5*x1)/m ))*5 / c;
```

```
I2 = g*m*(1 - exp(-c*(5 + 5*x2)/m ))*5 / c;
I3 = g*m*(1 - exp(-c*(5 + 5*x3)/m ))*5 / c;
I4 = g*m*(1 - exp(-c*(5 + 5*x4)/m ))*5 / c;
I = c1*I1 + c2 * I2 + c3 * I3 + c4 * I4;
disp(I,"integral by four point method =")
x1 = -0.906179846;
x2 = -0.538469310;
x3 = 0;
x4 = 0.538469310;
x5 = 0.906179846
c1 = 0.2369269;
c2 = 0.4786287;
c3 = 0.5688889;
c4 = 0.4786287;
c5 = 0.2369269;
I1 = g*m*(1 - exp(-c*(5 + 5*x1)/m ))*5 / c;
I2 = g*m*(1 - exp(-c*(5 + 5*x2)/m ))*5 / c;
I3 = g*m*(1 - exp(-c*(5 + 5*x3)/m ))*5 / c;
I4 = g*m*(1 - exp(-c*(5 + 5*x4)/m ))*5 / c;
I5 = g*m*(1 - exp(-c*(5 + 5*x5)/m ))*5 / c;
I = c1*I1 + c2 * I2 + c3 * I3 + c4 * I4 + c5 * I5;
disp(I,"integral by five point method =")
```

Scilab code Exa 22.6 Evaluation of improper integral

```
//clc()
//N(x) = (int(-infinity,-2,exp(-(x^2)/2)) + int(-2,1,exp(-(x^2)/2)))/(2*pi)^0.5
//first integral can be solved as
//int(-infinity,-2,exp(-(x^2)/2)) = int(-0.5,0,exp(-1/(2*t^2))/t^2)
h = 1/8;
//int(-0.5,0,exp(-1/(2*t^2))/t^2) = h*(f(x-7/16) + f(x-5/16) + f(x-3/16) + f(x-1/16))
t1 = -7/16;
```

```
t2 = -5/16;
t3 = -3/16;
t4 = -1/16;
m1 = exp(-1/(2*t1^2))/t1^2;
m2 = exp(-1/(2*t2^2))/t2^2;
m3 = exp(-1/(2*t3^2))/t3^2;
m4 = exp(-1/(2*t4^2))/t4^2;
I1 = h*(m1 + m2 + m3 + m4);
disp(I1,"the value of first integral = ")
//simpsons 1/3rd sule is applied for the second
    integral
h1 = 0.5;
x1 = -2;
x2 = -1.5;
x3 = -1;
x4 = -0.5;
x5 = 0;
x6 = 0.5;
x7 = 1;
n1 = exp(-(x1^2)/2);
n2 = exp(-(x2^2)/2);
n3 = exp(-(x3^2)/2);
n4 = exp(-(x4^2)/2);
n5 = exp(-(x5^2)/2);
n6 = exp(-(x6^2)/2);
n7 = exp(-(x7^2)/2);
I2 =(1-(-2)) * (n1 + 4 *(n2 + n4 + n6) + 2*(n3 + n5)
     + n7)/(18);
disp(I2,"The value of second integral = ")
f = (I1 + I2)/(2 * %pi)^0.5;
disp(f,"Therefore the final result can be computed
    as ")
N = 0.8413;
e = (N - f) * 100 / N;
disp("%",e,"error = ")
```

Chapter 23

Numerical differentiation

Scilab code Exa 23.1 High accuracy numerical differentiation formulas

```
//clc()
//f(x) = −0.1*x^4 − 0.15*x^3 − 0.5 * x^2 − 0.25 *x + 1.2
h = 0.25;
t = -0.9125;
x = 0:h:1;
disp(x,"x = ")
fx = -0.1*x^4 - 0.15*x^3 - 0.5 * x^2 - 0.25 *x + 1.2;
disp(fx,"f(x) = ")
fd = (- fx(5) + 4*fx(4) - 3 * fx(3))/(2 * h);
efd = (t - fd) * 100 / t;
disp(fd,"by forward difference")
disp("%",efd,"error in forward difference method = ")
bd = (3 * fx(3) - 4 * fx(2) + fx(1))/ (2*h);
ebd = (t - bd) * 100 / t;
disp(bd,"by backward difference")
disp("%",ebd,"error in backward difference method = ")
cdm = (-fx(5) + 8*(fx(4)) -8*fx(2) + fx(1) ) / (12*h
```

```
);
ecdm = (t - cdm) * 100 / t;
disp(cdm,"by central difference")
disp("%",ecdm,"error in central difference method = ")
```

Scilab code Exa 23.2 Richardson extrapolation

```
//clc()
//f(x) = -0.1*x^4 - 0.15*x^3 - 0.5 * x^2 - 0.25 *x + 1.2
h = 0.5;
t = -0.9125;
x = 0:h:1;
disp(x,"x with h = 0.5 is")
fx = -0.1*x^4 - 0.15*x^3 - 0.5 * x^2 - 0.25 *x + 1.2;
disp(fx,"f(x) with h = 0.5 is")
cdm = (fx(3) - fx(1))/ 1;
ecdm = (t - cdm) * 100 / t;
disp(cdm,"by central difference ( h = 0.5 ) ")
disp("%",ecdm,"error in central difference method ( h = 0.5 ) = ")
h1 = 0.25;
x1 = 0:h1:1;
disp(x1,"x with h = 0.25 is")
fx1 = -0.1*x1^4 - 0.15*x1^3 - 0.5 * x1^2 - 0.25 *x1 + 1.2;
disp(fx1,"fx with h = 0.25 is")
cdm1 = (fx1(4) - fx1(2))/ (2*h1);
ecdm1 = (t - cdm1) * 100 / t;
disp(cdm1,"by central difference ( h = 0.25 ) ")
disp("%",ecdm1,"error in central difference method ( h = 0.25 ) = ")
D = 4 * cdm1 /3 - cdm / 3;
```

```
disp(D,"improved estimate =")
```

Scilab code Exa 23.3 Differentiating unequally spaced data

```
//clc()
//q(z = 0) = −k*p*C*(dT/dz)/(z = 0)
k = 3.5 * 10^ - 7;//m^2/s
p = 1800;//kg/m^3
C = 840; //(J/(kg.C))
x = 0;
fx0 = 13.5;
fx1 = 12;
fx2 = 10;
x0 = 0;
x1 = 1.25;
x2 = 3.75;
dfx = fx0 *(2*x - x1 - x2)/((x0 - x1)*(x0 - x2)) +
    fx1 *(2*x - x0 - x2)/((x1 - x0)*(x1 - x2)) + fx2
    *(2*x - x1 - x0)/((x2 - x1)*(x2 - x0));
disp("C/cm",dfx,"(dT/dz) = ")
q = - k * p *C * dfx*100;
disp("W/m^2",q,"q(z = 0) =")
```

Scilab code Exa 23.4 Integration and Differentiation

```
clc;
clear;
function y=f(x)
    y=0.2+25*x-200*x^2+675*x^3-900*x^4+400*x^5
endfunction
a=0;
b=0.8;
Q=intg(0,0.8,f);
```

```
disp(Q,"Q=")
x=[0 0.12 0.22 0.32 0.36 0.4 0.44 0.54 0.64 0.7
   0.8];
y=f(x);
integral=inttrap(x,y)
disp(integral,"intergral=")
disp(diff(x),"diff(x)=")
d=diff(y)./diff(x);
disp(d,"d=")
```

Scilab code Exa 23.5 Integrate a function

```
clc;
clear;
function y=f(x)
    y=0.2+25*x-200*x^2+675*x^3-900*x^4+400*x^5
endfunction
a=0;
b=0.8;
Qt=1.640533;
Q=intg(0,0.8,f);
disp(Q,"Computed=")
disp(abs(Q-Qt)*100/Qt,"Error  estimate=")
```

Chapter 25

Runga Kutta methods

Scilab code Exa 25.1 Eulers method

```
//clc()
//dy/dx = -2*x^3 + 12*x^2 - 20*x + 8.5
//therefore, y = -0.5*x^4 + 4*x^3 - 10*x^2 + 8.5 + c
x1 = 0;
y1 = 1;
h = 0.5;
c =-(-0.5*x1^4 + 4*x1^3 - 10*x1^2 + 8.5*x1 - y1);
x = 0:0.5:4;
disp(x,"x = ")
y = -0.5*x^4 + 4*x^3 - 10*x^2 + 8.5*x + c;
disp(y,"true values of y = ")
fxy = -2*x^3 + 12*x^2 - 20*x + 8.5;
y2(1) = y(1);
e(1) = (y(1) - y2(1)) * 100 / y(1);
for i = 2:9
    y2(i) = y2(i-1) + fxy(i-1)*h;
    e(i) = (y(i) - y2(i))*100/y(i);
end
disp(y2,"y by euler method =")
disp(e,"error =")
```

Scilab code Exa 25.2 Taylor series estimate for error by eulers method

```
//clc()
//f(x,y) = dy/dx = -2*x^3 + 12*x^2 - 20*x + 8.5
//f'(x,y) = -6*x^2 + 24*x - 20
//f"(x,y) = -12*x + 24
//f"'(x,y) = -12
x = 0;
Et2 = (-6*x^2 + 24*x - 20) * 0.5^2 / 2;
Et3 = (-12*x + 24) * (0.5)^3 / 6;
Et4 = (-12) *(0.5 ^ 4) / 24;
Et = Et2 + Et3 + Et4;
disp(Et,"Total truncation error =")
```

Scilab code Exa 25.3 Effect of reduced step size on Eulers method

```
//clc()
//dy/dx = -2*x^3 + 12*x^2 - 20*x + 8.5
//therefore , y = -0.5*x^4 + 4*x^3 - 10*x^2 + 8.5 + c
x1 = 0;
y1 = 1;
h = 0.25;
c =-(-0.5*x1^4 + 4*x1^3 - 10*x1^2 + 8.5*x1 - y1);
x = 0:h:4;
disp(x,"x = ")
y = -0.5*x^4 + 4*x^3 - 10*x^2 + 8.5*x + c;
disp(y,"true values of y = ")
fxy = -2*x^3 + 12*x^2 - 20*x + 8.5;
y2(1) = y(1);
e(1) = (y(1) - y2(1)) * 100 / y(1);
for i = 2:17
    y2(i) = y2(i-1) + fxy(i-1)*h;
```

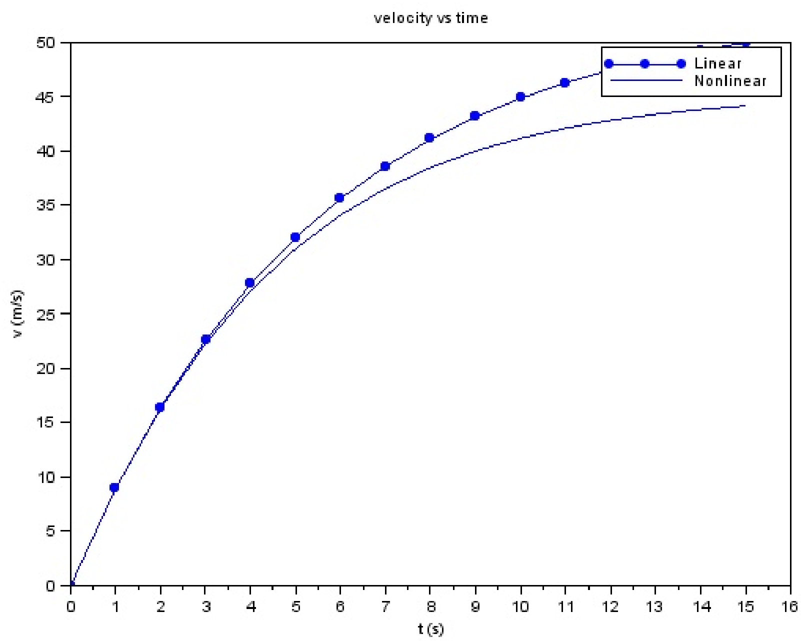

Figure 25.1: Solving ODEs

```
    e(i) = (y(i) - y2(i))*100/y(i);
end
disp(y2,"y by euler method =")
disp(e,"error =")
```

Scilab code Exa 25.4 Solving ODEs

```
clc;
clear;
m=68.1;
g=9.8;
```

```
c=12.5;
a=8.3;
b=2.2;
vmax=46;
function yp=f(t,v)
    yp=g-c*v/m;
endfunction
v0=0;
t=0:15;
x=ode(v0,0,t,f);
disp(x)
plot(t,x,'.-')

function yp=f1(t,v)
    yp=g-(c/m)*(v+a*(v/vmax)^b)
endfunction
x1=ode(v0,0,t,f1);
plot(t,x1)
xtitle("velocity vs time","t (s)","v (m/s)")
h1=legend(["Linear";"Nonlinear"])
```

Scilab code Exa 25.5 Heuns method

```
//clc()
//y' = 4*exp(0.8*x) - 0.5*y
//y = 4*(exp(0.8*x) - exp(-0.5*x))/1.3 + 2*exp(-0.5*
   x)
x = 0:1:4;
disp(x)
x1 = 0;
y1 = 2;
y2(1) = y1;
for i = 1:5
    y(i) = 4*(exp(0.8*x(i)) - exp(-0.5*x(i)))/1.3 +
       2*exp(-0.5*x(i));
```

```
    dy(i) = 4*exp(0.8*x(i)) - 0.5*y2(i);
    y2(i + 1) = y2(i) + dy(i);
    if i>1 then
        m(i) = (dy(i) + dy(i-1))/2;
        y2(i) = y2(i-1) + m(i);
        dy(i) = 4*exp(0.8*x(i)) - 0.5*y2(i);
    end
    e(i) = (y(i) - y2(i)) * 100 / y(i);
end
disp(y2(1:5),"By heuns method(1 iteration)")
disp("%",e(1:5),"error = ")
```

Scilab code Exa 25.6 Comparison of various second order RK 4 method

```
//clc()
//f'(x,y) = -2*x^3 + 12*x^2 -20*x + 8.5
//f(x,y) = -x^4 / 2 + 4*x^3 - 10*x^2 + 8.5*x + 1
h = 0.5;
x = 0:h:4;
y1 = -x^4 / 2 + 4*x^3 - 10*x^2 + 8.5*x + 1;
y(1) = 1;
disp(x,"x =")
disp(y1,"true value of y =")
for i = 1:8
    k1(i) = -2*x(i)^3 + 12*x(i)^2 -20*x(i) + 8.5;
    x1(i) = x(i) + h/2;
    k2(i) = -2*x1(i)^3 + 12*x1(i)^2 -20*x1(i) + 8.5;
    y(i+1) = y(i) + k2(i)*h;
    e(i) = (y1(i) - y(i))*100/y1(i);
end
disp(y(1:9),"y by midpoint method")
disp(e,"error = ")
for i = 1:8
    k1(i) = -2*x(i)^3 + 12*x(i)^2 -20*x(i) + 8.5;
    x(i) = x(i) + 3*h/4;
```

```
    k2(i) = -2*x(i)^3 + 12*x(i)^2 -20*x(i) + 8.5;
    y(i+1) = y(i) + (k1(i)/3 + 2*k2(i)/3)*h;
    e(i) = (y1(i) - y(i))*100/y1(i);
end
disp(y(1:9),"y by second order Ralston RK")
disp(e,"error = ")
```

Scilab code Exa 25.7 `Classical fourth order RK method`

```
//clc()
//f'(x,y) = -2*x^3 + 12*x^2 -20*x + 8.5
//f(x,y) = -x^4 / 2 + 4*x^3 - 10*x^2 + 8.5*x + 1
h = 0.5;
x = 0:h:4;
y1 = -x^4 / 2 + 4*x^3 - 10*x^2 + 8.5*x + 1;
y(1) = 1;
for i=1:8
     k1(i) = -2*x(i)^3 + 12*x(i)^2 -20*x(i) + 8.5;
    x1(i) = x(i) + h/2;
    k2(i) = -2*x1(i)^3 + 12*x1(i)^2 -20*x1(i) + 8.5;
    k3(i) = -2*x1(i)^3 + 12*x1(i)^2 -20*x1(i) + 8.5;
    x2(i) = x(i) + h;
    k4(i) = -2*x2(i)^3 + 12*x2(i)^2 -20*x2(i) + 8.5;
    y(i+1) = y(i) + (k1(i)+2*k2(i)+2*k3(i)+k4(i))*h
       /6;
    e(i) = (y1(i) - y(i))*100/y1(i);
end
disp("f(x,y) = -2*x^3 + 12*x^2 -20*x + 8.5")
disp(y(1:9),"y by fourth order Ralston RK")
disp("f(x,y) = 4*exp(0.8*x) - 0.5*y")
x = 0:h:0.5;
y(1) = 2;
k1 = 4*(exp(0.8*x(1)))-0.5*y(1);
x1 = x(1) + 0.5*h;
y1 = y(1) + 0.5*k1*h;
```

```
k2 = 4*exp(0.8*x1) - 0.5*y1;
y2 = y(1) + 0.5*k2*h;
k3 = 4*exp(0.8*x1) - 0.5*y2;
x1 = x(1) + h;
y1 = y(1) + k3*h;
k4 = 4*exp(0.8*x1) - 0.5*y1;
y(2) = y(1) + (k1+2*k2+2*k3+k4)*h/6;
disp(y(1:2),"y = ")
```

Scilab code Exa 25.8 Comparison of Runga Kutta methods

```
//clc()
disp("f(x,y) = 4*exp(0.8*x) - 0.5*y")
h = 1;
x = 0:h:4;
y(1) = 2;
for i = 1:5
    k1(i) = 4*(exp(0.8*x(i)))-0.5*y(i);
    x1 = x(i) + h;
    y1 = y(i) + k1(i)*h;
    k2(i) = 4*(exp(0.8*x1))-0.5*y1;
    y(i+1) = y(i) + (k1(i)/2 + k2(i)/2)*h;
end
disp(y(1:5),"y(second order RK method) = ")
for i = 1:5
    k1(i) = 4*(exp(0.8*x(i)))-0.5*y(i);
    x1 = x(i) + 0.5*h;
    y1 = y(i) + 0.5*h*k1(i);
    k2(i) = 4*(exp(0.8*x1))-0.5*y1;
    x1 = x(i) + h;
    y1 = y(i) -k1(i)*h + 2*k2(i)*h;
    k3(i) = 4*(exp(0.8*x1))-0.5*y1;
    y(i+1) = y(i) + (k1(i) + 4*k2(i) + k3(i))*h/6;
end
disp(y(1:5),"y(third order RK method) = ")
```

```
for i = 1:5
k1(i) = 4*(exp(0.8*x(i)))-0.5*y(i);
x1 = x(i) + 0.5*h;
y1 = y(i) + 0.5*k1(i)*h;
k2(i) = 4*exp(0.8*x1) - 0.5*y1;
y2 = y(i) + 0.5*k2(i)*h;
k3(i) = 4*exp(0.8*x1) - 0.5*y2;
x1 = x(i) + h;
y1 = y(i) + k3(i)*h;
k4(i) = 4*exp(0.8*x1) - 0.5*y1;
y(i+1) = y(i) + (k1(i)+2*k2(i)+2*k3(i)+k4(i))*h/6;
end
disp(y(1:5),"y(fourth order RK method) = ")
for i = 1:5

k1(i) = 4*(exp(0.8*x(i)))-0.5*y(i);
x1 = x(i) + 0.25*h;
y1 = y(i) + 0.25*k1(i)*h;
k2(i) = 4*exp(0.8*x1) - 0.5*y1;
y2 = y(i) + 0.125*k2(i)*h + 0.125*k1(i)*h;
k3(i) = 4*exp(0.8*x1) - 0.5*y2;
x1 = x(i) + 0.5*h;
y1 = y(i) -0.5*k2(i)*h + k3(i)*h;
k4(i) = 4*exp(0.8*x1) - 0.5*y1;
x1 = x(i) + 0.75*h;
y1 = y(i) + 3*k1(i)*h/16 + 9*k4(i)*h/16;
k5(i) = 4*exp(0.8*x1) - 0.5*y1;
x1 = x(i) + h;
y1 = y(i) - 3*k1(i)*h/7 + 2*k2(i)*h/7 + 12*k3(i)*h/7 - 12*k4(i)*h/7 + 8*k5(i)*h/7;
k6(i) = 4*exp(0.8*x1) - 0.5*y1;
y(i+1) = y(i) + (7*k1(i)+32*k3(i)+12*k4(i) + 32*k5(i) + 7*k6(i))*h/90;
end
disp(y(1:5),"y(fifth order RK method)")
```

Scilab code Exa 25.9 Solving systems of ODE using Eulers method

```
//clc()
//dy1/dx = -0.5*y1
//dy2/dx = 4 - 0.3*y2 - 0.1*y1
x1 = 0;
h =0.5;
y1(1) = 4;
y2(1) = 6;
x = 0:h:2;
for i = 2:5
    y1(i) = y1(i-1)  -0.5*y1(i-1)*h;
    y2(i) = y2(i-1) + (4 - 0.3*y2(i-1) - 0.1*y1(i-1)
       )*h;
end
disp(x,"x = ")
disp(y1,"y1 = ")
disp(y2,"y2 = ")
```

Scilab code Exa 25.10 Solving systems of ODE using RK 4 method

```
//clc()
//dy1/dx = -0.5*y1
//dy2/dx = 4 - 0.3*y2 - 0.1*y1
x1 = 0;
h =0.5;
y1(1) = 4;
y2(1) = 6;
x = 0:h:2;
for i = 1:5
    k11(i) = -0.5*y1(i);
    k12(i) = 4 - 0.3*y2(i) - 0.1*y1(i);
```

```
    y11 = y1(i) + k11(i) * h/2;
    y21 = y2(i) + k12(i) * h/2;
    k21(i) = -0.5*y11;
    k22(i) = 4 - 0.3*y21 - 0.1*y11;
    y11 = y1(i) + k21(i) * h/2;
    y21 = y2(i) + k22(i) * h/2;
    k31(i) = -0.5*y11;
    k32(i) = 4 - 0.3*y21 - 0.1*y11;
    y11 = y1(i) + k31(i) * h;
    y21 = y2(i) + k32(i) * h;
    k41(i) = -0.5*y11;
    k42(i) = 4 - 0.3*y21 - 0.1*y11;
    y1(i+1) = y1(i) + ( k11(i) + 2*k21(i) + 2*k31(i)
        + k41(i) )*h / 6;
    y2(i+1) = y2(i) + ( k12(i) + 2*k22(i) + 2*k32(i)
        + k42(i) ) *h/ 6;
end
disp("using fourth order RK method")
disp(x,"x =")
disp(y1(1:5),"y1 = ")
disp(y2(1:5),"y2 = ")
```

Scilab code Exa 25.11 Solving systems of ODEs

```
clc;
clear;
function yp=f(x,y)
    yp=[y(2);-16.1*y(1)];
endfunction
x=0:0.1:4
y0=[0.1 0];
sol=ode(y0,0,x,f);
count=1;
disp(sol)
for i=1:2:81
```

```
    a(count)=sol(i);
    b(count)=sol(i+1);
    count=count+1;
end
plot(x,a)
plot(x,b,".-")
h1=legend(["y1,y3","y2,y4"])
xtitle("y vs x","x","y")
function yp=g(x,y)
    yp=[y(2);-16.1*sin(y(1))];
endfunction
sol=ode(y0,0,x,g);
count=1;
disp(sol)
for i=1:2:81
    a(count)=sol(i);
    b(count)=sol(i+1);
    count=count+1;
end
plot(x,a)
plot(x,b,".-")
clf();
y0=[%pi/4 0];
sol=ode(y0,0,x,f);
count=1;
disp(sol)
for i=1:2:81
    a(count)=sol(i);
    b(count)=sol(i+1);
    count=count+1;
end
plot(x,a)
plot(x,b,".-")

xtitle("y vs x","x","y")
sol=ode(y0,0,x,g);
count=1;
disp(sol)
```

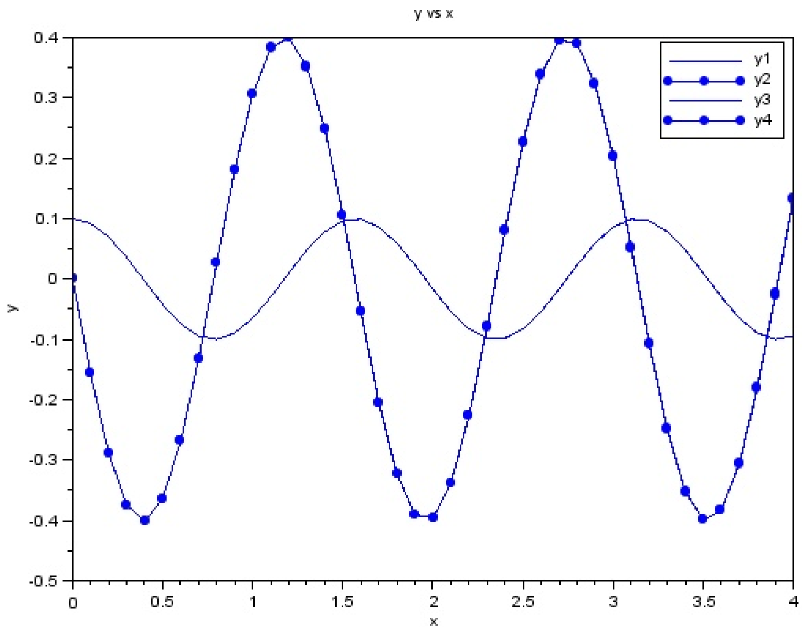

Figure 25.2: Solving systems of ODEs

```
for i=1:2:81
    a(count)=sol(i);
    b(count)=sol(i+1);
    count=count+1;
end
plot(x,a,'o-')
plot(x,b,"x-")
h1=legend(["y1","y2","y3","y4"])
```

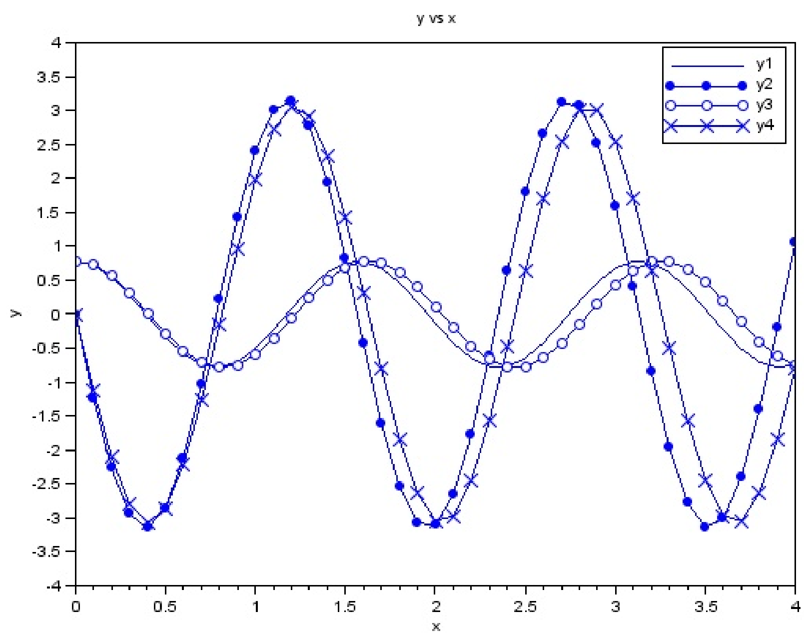

Figure 25.3: Solving systems of ODEs

Scilab code Exa 25.12 Adaptive fourth order RK method

```
//clc()
disp("f(x,y) = 4*exp(0.8*x) - 0.5*y")
//f'(x,y) = 4*exp(0.8*x) - 0.5*y
x = 0:2:2;
y(1) = 2;
h =2;
t = 14.84392;
k1 = 4*exp(0.8*x(1)) - 0.5*y(1);
x1 = x(1) + h/2;
y1 = y(1) + k1*h/2;
k2 = 4*exp(0.8*x1) - 0.5*y1;
x1 = x(1) + h/2;
y1 = y(1) + k2*h/2;
k3 = 4*exp(0.8*x1) - 0.5*y1;
x1 = x(1) + h;
y1 = y(1) + k3*h;
k4 = 4*exp(0.8*x1) - 0.5*y1;
y(2) = y(1) + (k1 + 2*k2 + 2*k3 + k4)*h/6;
e = (t - y(2))/(t);
disp(y(1:2),"y by h = 2 is")
disp(e,"error = ")
h = 1;
x = 0:h:2;
for i=1:3
    k1(i) = 4*exp(0.8*x(i)) - 0.5*y(i);
    x1 = x(i) + h/2;
    y1 = y(i) + k1(i)*h/2;
    k2(i) = 4*exp(0.8*x1) - 0.5*y1;
    x1 = x(i) + h/2;
    y1 = y(i) + k2(i)*h/2;
    k3(i) = 4*exp(0.8*x1) - 0.5*y1;
    x1 = x(i) + h;
    y1 = y(i) + k3(i)*h;
    k4(i) = 4*exp(0.8*x1) - 0.5*y1;
    y(i+1) = y(i) + (k1(i) + 2*k2(i) + 2*k3(i) + k4(
       i))*h/6;
```

```
end
e = (t - (y(3)))/t;
disp(y(1:3),"y by h = 1 is")
disp(e,"error = ")
```

Scilab code Exa 25.13 Runga kutta fehlberg method

```
//clc()
disp("f(x,y) = 4*exp(0.8*x) - 0.5*y")
//f'(x,y) = 4*exp(0.8*x) - 0.5*y
h =2;
x = 0:h:2;
y(1) = 2;
t = 14.84392;
k1 = 4*exp(0.8*x(1)) - 0.5*y(1);
x1 = x(1) + h/5;
y1 = y(1) + k1*h/5;
k2 = 4*exp(0.8*x1) - 0.5*y1;
x1 = x(1) + 3*h/10;
y1 = y(1) + 3*k1*h/40 + 9*k2*h/40;
k3 = 4*exp(0.8*x1) - 0.5*y1;
x1 = x(1) + 3*h/5;
y1 = y(1) + 3*k1*h/10 - 9*k2*h/10 + 6*k3*h/5;
k4 = 4*exp(0.8*x1) - 0.5*y1;
x1 = x(1) + h;
y1 = y(1) -11*k1*h/54 + 5*k2*h/2 - 70*k3*h/27 + 35*
    k4*h/27;
k5 = 4*exp(0.8*x1) - 0.5*y1;
x1 = x(1) + 7*h/8;
y1 = y(1) + 1631*k1*h/55296 + 175*k2*h/512 + 575*k3*
    h/13824 + 44275*k4*h/110592 +253*k5*h/4096;
k6 = 4*exp(0.8*x1) - 0.5*y1;
y1 = y(1) + (37*k1/378 + 250*k3/621 + 125*k4/594 +
    512*k6/1771)*h;
y2 = y(1) + (2825*k1/27648 + 18575*k3/48384 + 13525*
```

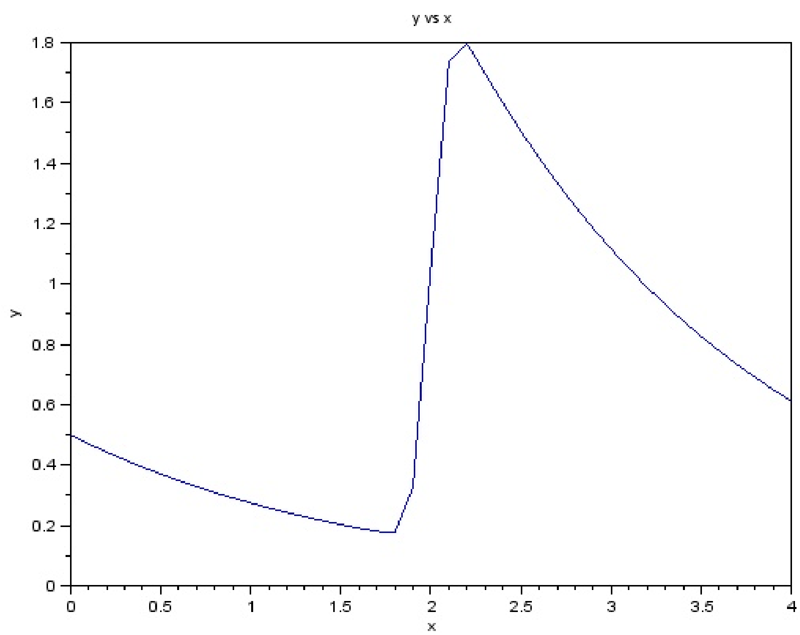

Figure 25.4: Adaptive Fourth order RK scheme

```
    k4/55296 + 277*k5/14336 + k6/4)*h;
disp(y1,"y ( fourth order prediction ) = ")
disp(y2,"y ( fifth order prediction ) = ")
```

Scilab code Exa 25.14 `Adaptive Fourth order RK scheme`

```
clc;
clear;
function yp=f(x,y)
    yp=10*exp(-(x-2)^2/(2*(0.075^2)))-0.6*y
endfunction
```

```
x=0:0.1:4
y0=0.5;
sol=ode(y0,0,x,f);
plot(x,sol)
xtitle("y vs x","x","y")
```

Chapter 26

Stiffness and multistep methods

Scilab code Exa 26.1 `Explicit and Implicit Euler`

```
clc;
clear;
function yp=f(t,y)
    yp=-1000*y+3000-2000*exp(-t)
endfunction
y0=0;
//explicit euler
h1=0.0005;
y1(1)=y0;
count=2;
t=0:0.0001:0.006
for i=0:0.0001:0.0059
    y1(count)=y1(count-1)+f(i,y1(count-1))*h1
    count=count+1;
end
plot(t,y1)
h2=0.0015;
y2(1)=y0;
count=2;
t=0:0.0001:0.006
for i=0:0.0001:0.0059
```

```
    y2(count)=y2(count-1)+f(i,y2(count-1))*h2
    count=count+1;
end
plot(t,y2,'.-')
xtitle("y vs t","t","y")
h=legend(["h-0.0005","h=0.0015"])
clf();
//implicit order
h3=0.05;
y3(1)=y0;
count=2;
t=0:0.01:0.4
for j=0:0.01:0.39
    y3(count)=(y3(count-1)+3000*h3-2000*h3*exp(-(j
       +0.01)))/(1+1000*h3)
    count=count+1;
end

plot(t,y3)
xtitle("y vs t","t","y")
```

Scilab code Exa 26.2 Non self starting Heun method

```
//clc()
disp("f(x,y) = 4*exp(0.8*x) - 0.5*y")
//f'(x,y) = 4*exp(0.8*x) - 0.5*y
h = 1;
x=0:h:4;
y(1) = 2;
x1 = -1;
y1 = -0.3929953;
```

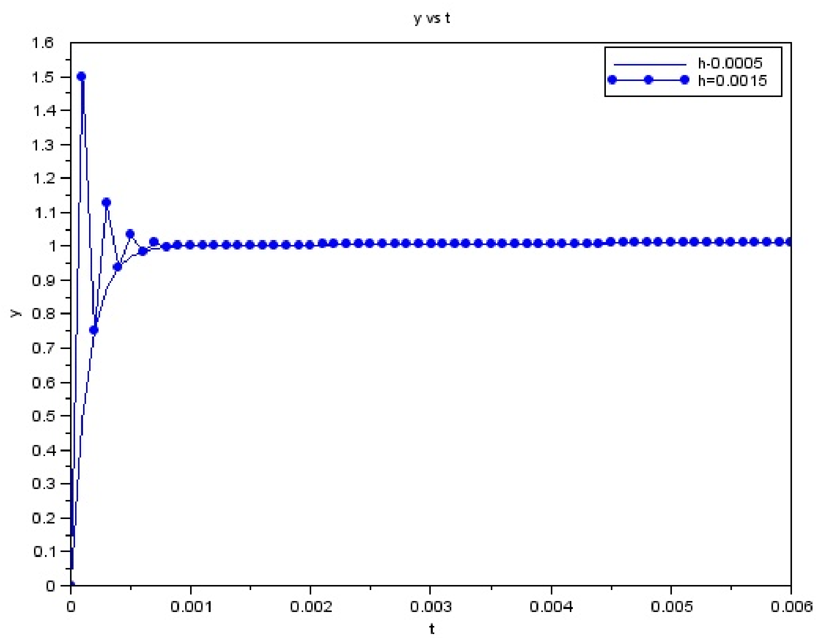

Figure 26.1: Explicit and Implicit Euler

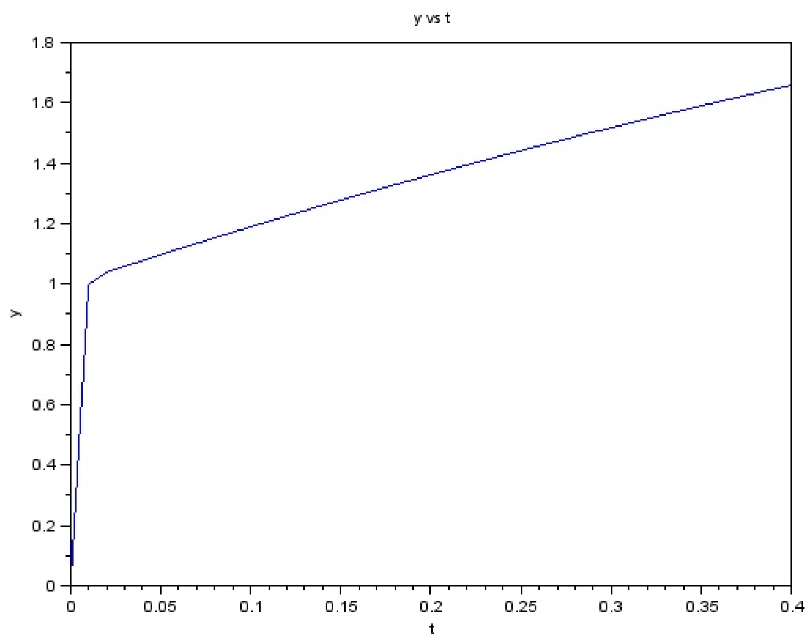

Figure 26.2: Explicit and Implicit Euler

```
y10 = y1 + (4*exp(0.8*x(1)) - 0.5*y(1))*2;
y11 = y(1) + (4*exp(0.8*x(1)) - 0.5*y(1) + 4*exp
    (0.8*x(2)) - 0.5*y10)*h/2;
y12 = y(1) + (3 + 4*exp(0.8*x(2)) - 0.5*y11)*h/2;
t = 6.360865;
y20 = y(1) + (4*exp(0.8*x(2)) - 0.5*t) *2;
y21 = t + (4*exp(0.8*x(2)) - 0.5*t + 4*exp(0.8*x(3))
     - 0.5*y20)*h/2;
disp(y21,"the first corrector yields y = ")
t = 14.84392
e = (t - y21)*100/t;
disp("%",e," error = ")
```

Scilab code Exa 26.3 Estimate of per step truncation error

```
//clc()
x1 = 1;
x2 = 2;
y1 = 6.194631;
y2 = 14.84392;
y10 = 5.607005;
y11 = 6.360865;
y20 = 13.44346;
y21 = 15.30224;
Ec1 = -(y11 - y10)/5;
disp(Ec1,"Ec (x = 1) = ")
e1 = y1 - y11;
disp(e1,"true error (x = 1) = ")
Ec2 = -(y21 - y20)/5;
disp(Ec2,"Ec (x = 2) = ")
e2 = y2 - y21;
disp(e2,"true error (x = 2) = ")
```

Scilab code Exa 26.4 Effect of modifier on Predictor Corrector results

```
//clc()
//clc()
x0 = 0;
x1 = 1;
x2 = 2;
y1 = 6.194631;
y2 = 14.84392;
y10 = 5.607005;
y11 = 6.360865;
y1m = y11 - (y11 - y10)/5;
e = (y1 - y1m)*100/y1;
disp(y1m,"ym")
disp("%",e,"error = ")
y20 =2+(4*exp(0.8*x1) - 0.5*y1m)*2;
e2 = (y2 - y20)*100/y2;
disp(y20,"y20 = ")
disp("%",e2,"error = ")
y2o = y20 + 4* (y11 - y10)/5;
e2 = (y2 - y2o)*100/y2;
disp(y2o,"y20 = ")
disp("%",e2,"error = ")
y21 = 15.21178;
y23 = y21 - (y21 - y20)/5;
disp(y23,"y2 = ")
e3 = (y2 - y23)*100/y2;
disp("%",e3,"error = ")
```

Scilab code Exa 26.5 Milnes Method

```
//clc()
disp("f(x,y) = 4*exp(0.8*x) - 0.5*y")
//f'(x,y) = 4*exp(0.8*x) - 0.5*y
h = 1;
```

```
x = -3:h:4;
y(1) = -4.547302;
y(2) = -2.306160;
y(3) = -0.3929953;
y(4) = 2;
y1(4) = 2;
for  i = 4:7;
    y(i+1) = y(i-3) + 4*h*(2*(4*exp(0.8*x(i)) - 0.5*
        y(i)) - 4*exp(0.8*x(i-1)) + 0.5*y(i-1) +
        2*(4*exp(0.8*x(i-2)) - 0.5*y(i-2)))/3;
    y1(i+1) = y(i-1) + h*(4*exp(0.8*x(i-1)) - 0.5*y(
        i-1) +4 * (4*exp(0.8*x(i)) - 0.5*y(i)) + 4*
        exp(0.8*x(i+1)) - 0.5*y(i+1))/3;
end
disp(x(4:8),"x = ")
disp(y(4:8),"y0 = ")
disp(y1(4:8),"corrected  y1 = ")
```

Scilab code Exa 26.6 Fourth order Adams method

```
//clc()
disp("f(x,y) = 4*exp(0.8*x) - 0.5*y")
//f'(x,y) = 4*exp(0.8*x) - 0.5*y
h = 1;
x = -3:h:4;
y(1) = -4.547302;
y(2) = -2.306160;
y(3) = -0.3929953;
y(4) = 2;
m(4) = y(4);
for i =4:7
    y(i+1) = y(i) + h *(55* (4*exp(0.8*x(i)) - 0.5*y
        (i)) / 24 - 59 * (4*exp(0.8*x(i-1)) - 0.5*y(i
        -1)) / 24 + 37*(4*exp(0.8*x(i-2)) - 0.5*y(i
        -2))/24 - 9*(4*exp(0.8*x(i-3)) - 0.5*y(i-3))
```

```
/24);
    m(i+1) = y(i+1)
    y(i+1) = y(i) + h*(9*(4*exp(0.8*x(i+1)) - 0.5*y(i+1))/24 +19*(4*exp(0.8*x(i)) - 0.5*y(i))/24 - 5*(4*exp(0.8*x(i-1)) - 0.5*y(i-1))/24 + (4*exp(0.8*x(i-2)) - 0.5*y(i-2))/24);
end
disp(x(4:8),"x = ")
disp(m(4:8),"y0 = ")
disp(y(4:8),"y1 = ")
```

Scilab code Exa 26.7 stability of Milnes and Fourth order Adams method

```
//clc()
//dy/dx = -y
//y = exp(-x)
h = 0.5;
x = -1.5:h:10;
y(1) = exp(-x(1));
y(2) = exp(-x(2));
y(3) = exp(-x(3));
y(4) = 1;
m(4) = y(4);
for  i = 4:23;
    y(i+1) = y(i-3) + 4*h*(2*(-y(i)) + y(i-1) + 2*(-y(i-2)))/3;
    m(i+1) = y(i+1);
    y(i+1) = y(i-1) + h*(-y(i-1) +4 * (-y(i)) + (-y(i+1)))/3;
end
disp(x(4:24),"x = ")
disp(m(4:24),"y0(milnes method) = ")
disp(y(4:24),"corrected y1(milnes method) = ")
for i =4:23
    y(i+1) = y(i) + h *(55* (-y(i)) / 24 - 59 * (-y(
```

```
     i-1)) / 24 + 37*(-y(i-2))/24 - 9*(-y(i-3))
     /24);
   m(i+1) = y(i+1)
   y(i+1) = y(i) + h*(9*(-y(i+1))/24 +19*(-y(i))/24
      - 5*(-y(i-1))/24 + (-y(i-2))/24);
end
disp(m(4:23),"y0(fourth order adams method) = ")
disp(y(4:23),"y1(fourth order adams method) = ")
```

Chapter 27

Boundary Value and Eigen Value problems

Scilab code Exa 27.1 The shooting method

```
//clc()
l = 10;//m
h = 0.01;//m^(-2)
Ta = 20;
T0 = 40;
T10 = 200;
//dT/dx = z
//dz/dx = h'(T-Ta)
// we use z = 10 as initial guess and integrating above equations simultaneously, solving using RK4 method, we get T10 = 168.3797
// similarly, with z = 20, T10 = 285.898
z01 = 10;
z02 = 20;
T10a = 168.3797;
T10b = 285.898;
z0 = z01 + (z02 - z01)*(T10 - T10a)/(T10b - T10a);
disp(z0,"z0 = ")
disp("this value of z can be used to get the correct
```

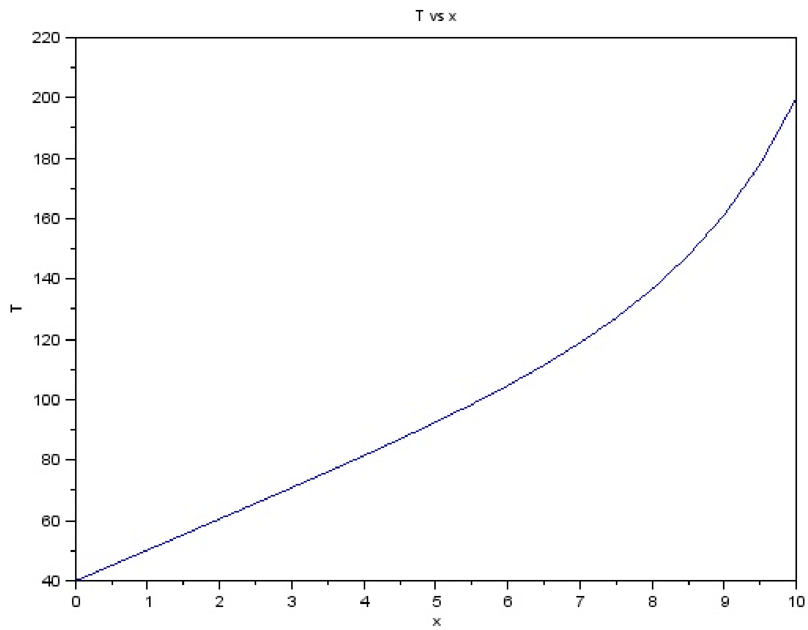

Figure 27.1: The shooting method for non linear problems

```
value")
```

Scilab code Exa 27.2 `The shooting method for non linear problems`

```
//clc()
z(1) = 10.0035;
T(1) = 40;
Ta = 20;
h = 0.5;
for i = 1:20
    k11(i) = z(i);
```

```
    k12(i) = 5*10^-8*(T(i) - Ta)^4;
    z1 = z(i) + h/2;
    T1 = T(i) + h/2;
    k21(i) = z1;
    k22(i) = 5*10^-8*(T1 - Ta)^4;
    z1 = z(i) + h/2;
    T1 = T(i) + h/2;
    k31(i) = z1;
    k32(i) = 5*10^-8*(T1 - Ta)^4;
    z1 = z(i) + h;
    T1 = T(i) + h;
    k41(i) = z1;
    k42(i) = 5*10^-8*(T1 - Ta)^4;
    T(i+1) = T(i) + ( k11(i) + 2* k21(i) + 2*k31(i)
       + k41(i))*h/6;
    z(i+1) = z(i) + ( k12(i) + 2* k22(i) + 2*k32(i)
       + k42(i))*h/6;
end
x=0:0.5:10;
plot(x,T(1:21))
xtitle("T vs x","x","T")
```

Scilab code Exa 27.3 Finite Difference Approximation

```
clc;
clear;
h=0.01;
delx=2;
x=2+h*delx^2;
a=[x -1 0 0;-1 x -1 0;0 -1 x -1;0 0 -1 x];
b=[40.8; 0.8; 0.8; 200.8];
T=linsolve(a,b);
disp("The temperature at the interior nodes:")
disp(abs(T))
```

Scilab code Exa 27.4 `Mass Spring System`

```
clc;
clear;
m1=40;//kg
m2=40;//kg
k=200;//N/m
sqw=poly(0,"s");
p=sqw^2-20*sqw+75;
r=roots(p);
f1=(r(1))^0.5;
f2=(r(2))^0.5;
Tp1=(2*%pi)/f1;
Tp2=(2*%pi)/f2;
//for first mode
disp("For first mode:")
disp(Tp1,"Period of oscillation:")
disp("A1=-A2")
disp("
    ____________________________________________________________
    ")
//for first mode
disp("For second mode:")
disp(Tp2,"Period of oscillation:")
disp("A1=A2")
```

Scilab code Exa 27.5 `Axially Loaded column`

```
clc;
clear;
E=10*10^9;//Pa
I=1.25*10^-5;//m^4
```

```
L=3;//m
for i=1:8
    p=i*%pi/L;
    P=i^2*(%pi)^2*E*I/(L^2*1000);
    disp(i,"n=")
    disp("m^-2",p,"p=")
    disp("kN",P,"P=")
    disp("
        -----------------------------------------------------------------------
        ")
end
```

Scilab code Exa 27.6 Polynomial Method

```
clc;
clear;
E=10*10^9;//Pa
I=1.25*10^-5;//m^4
L=3;//m
true=[1.0472 2.0944 3.1416 4.1888];
//part a
h1=3/2;
p=poly(0,"s")
a=2-h1^2*p^2;
x=roots(a);
e=abs(abs(x(1))-true(1))*100/true(1);
disp(x,"p=")
disp(e,"error=")
disp("
    -----------------------------------------------------------------"
    )
//part b
h2=3/3;
p=poly(0,"s")
a=(2-h2^2*p^2)^2 - 1;
```

```
x=roots(a);
e(1)=abs(abs(x(1))-true(2))*100/true(2);
e(2)=abs(abs(x(2))-true(1))*100/true(1);
disp(x,"p=")
disp(e,"error=")
disp("
    -----------------------------------------------------------
    ")
//part c
h3=3/4;
p=poly(0,"s")
a=(2-h3^2*p^2)^3 - 2*(2-h3^2*p^2);
x=roots(a);
e(1)=abs(abs(x(1))-true(3))*100/true(3);
e(2)=abs(abs(x(2))-true(2))*100/true(2);
e(3)=abs(abs(x(3))-true(1))*100/true(1);
disp(x,"p=")
disp(e,"error=")
disp("
    -----------------------------------------------------------
    ")


//part d
h4=3/5;
p=poly(0,"s")
a=(2-h4^2*p^2)^4 - 3*(2-h4^2*p^2)^2 + 1;
x=roots(a);
e(1)=abs(abs(x(1))-true(4))*100/true(4);
e(2)=abs(abs(x(2))-true(3))*100/true(3);
e(3)=abs(abs(x(3))-true(2))*100/true(2);
e(4)=abs(abs(x(4))-true(1))*100/true(1);
disp(x,"p=")
disp(e,"error=")
disp("
    -----------------------------------------------------------
    ")
```

Scilab code Exa 27.7 Power Method Highest Eigenvalue

```
clc;
clear;
a=[3.556 -1.668 0;-1.778 3.556 -1.778;0 -1.778
   3.556];
b=[1.778;0;1.778];
ea=100;
count=1;
while ea>0.1
    maxim=b(1);
    for i=2:3
        if abs(b(i))>abs(maxim) then
            maxim=b(i);
        end
    end
    eigen(count)=maxim;
    b=a*(b./maxim);
    if count==1 then
        ea=20;
        count=count+1;

    else
        ea=abs(eigen(count)-eigen(count-1))*100/abs(
           eigen(count));
        count=count+1;
    end
end
disp(eigen(count-1),"The largest eigen value")
```

Scilab code Exa 27.8 Power Method Lowest Eigenvalue

```
clc;
clear;
a=[3.556 -1.668 0;-1.778 3.556 -1.778;0 -1.778
   3.556];
b=[1.778;0;1.778];
ea=100;
count=1;
ai=inv(a);
while ea>4
    maxim=b(1);
    for i=2:3
        if abs(b(i))>abs(maxim) then
            maxim=b(i);
        end
    end
    eigen(count)=maxim;
    b=ai*(b./maxim);
    if count==1 then
        ea=20;
        count=count+1;

    else
        ea=abs(eigen(count)-eigen(count-1))*100/abs(
           eigen(count));
        count=count+1;
    end
end
disp((1/eigen(count-1))^0.5,"The smallest eigen
   value")
```

Scilab code Exa 27.9 Eigenvalues and ODEs

```
clc;
clear;
function yp=predprey(t,y)
```

```
    yp=[1.2*y(1)-0.6*y(1)*y(2);-0.8*y(2)+0.3*y(1)*y
       (2)];
endfunction
t=0:0.1:20;
y0=[2 1];
sol=ode(y0,0,t,predprey);
count=1;
for i=1:2:401
    x(count)=sol(i);
    z(count)=sol(i+1);
    count=count+1;
end

plot(t,x)
plot(t,z)
xtitle("y vs t", "t","y")
clf();
plot(x,z)
xtitle("space-space plot (y1 vs y2)", "y1","y2")
```

Scilab code Exa 27.10.a Stiff ODEs

```
clc;
clear;
function yp=vanderpol(t,y)
    yp=[y(2);1*(1-y(1)^2)*y(2)-y(1)]
endfunction
y0=[1 1];
t=0:0.1:20;
```

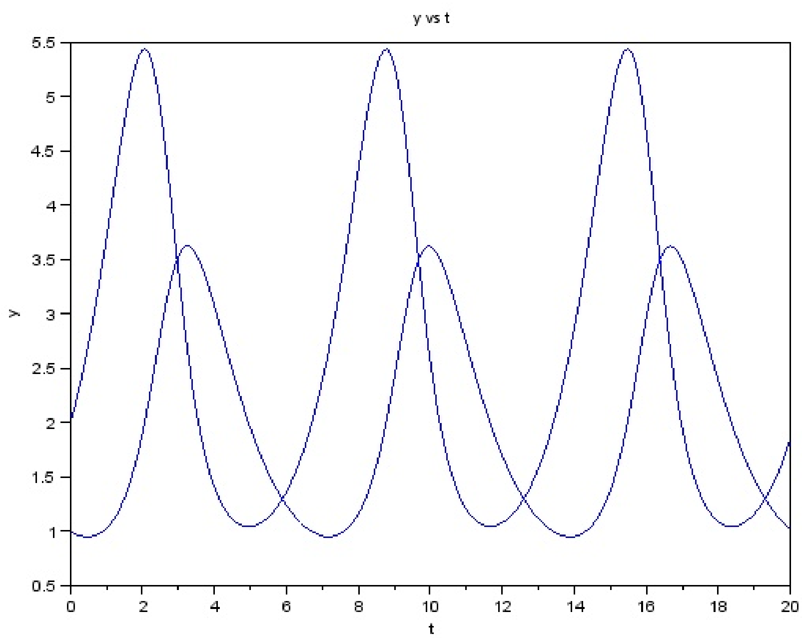

Figure 27.2: Eigenvalues and ODEs

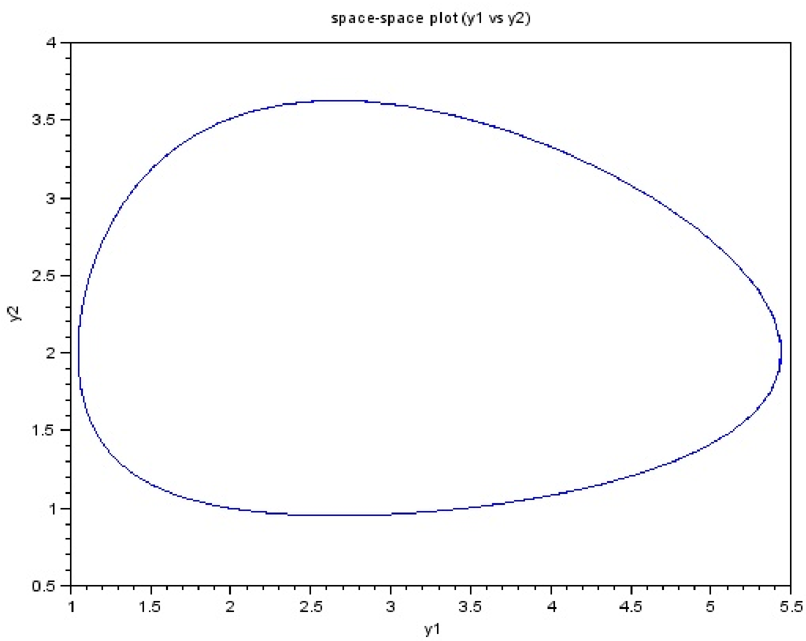

Figure 27.3: Eigenvalues and ODEs

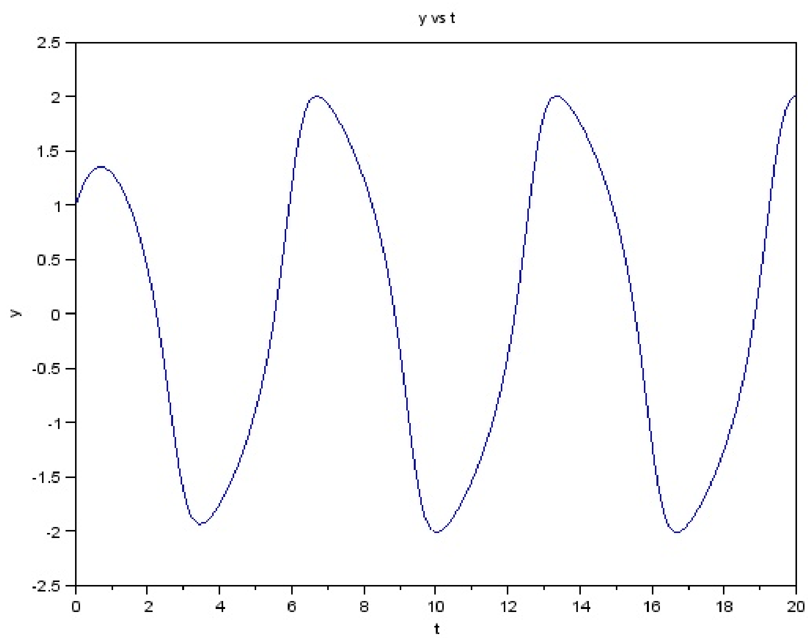

Figure 27.4: Stiff ODEs

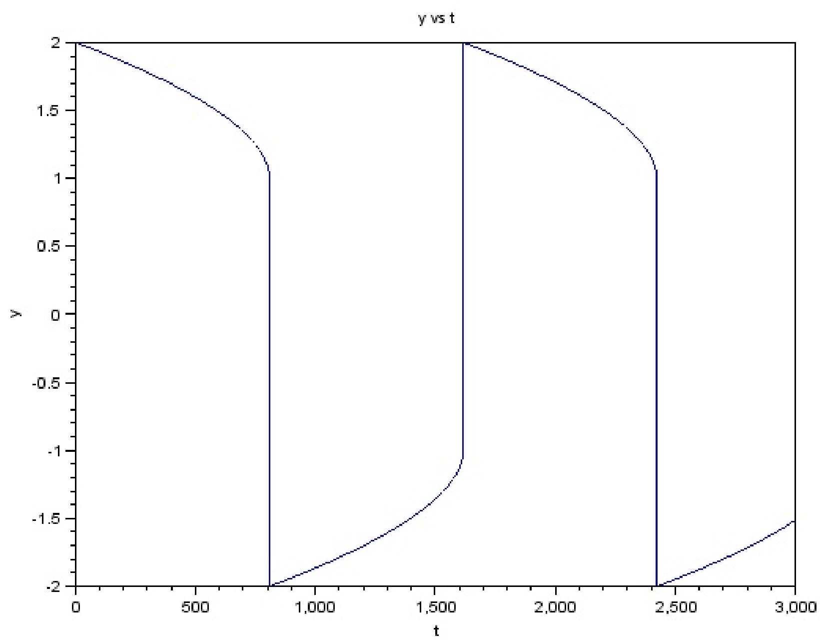

Figure 27.5: Stiff ODEs

```
sol=ode(y0,0,t,vanderpol);
count=1;
for i=1:2:401
    x(count)=sol(i);
    count=count+1;
end
plot(t,x)
xtitle("y vs t","t","y")
```

Scilab code Exa 27.10.b `Stiff ODEs`

```
clc;
clear;
clf();
function yp=vanderpol1(t,y)
    yp=[y(2);1000*(1-y(1)^2)*y(2)-y(1)]
endfunction
t=0:3000;
yo=[1 1];
sol=ode(yo,0,t,vanderpol1);
count=1;

for i=1:2:6001
    z(count)=sol(i)*-1;
    count=count+1;
end

plot(t,z)
xtitle("y vs t","t","y")
```

Scilab code Exa 27.11 Solving ODEs

```
clc;
clear;
function yp=predprey(t,y)
    yp=[1.2*y(1)-0.6*y(1)*y(2);-0.8*y(2)+0.3*y(1)*y
       (2)];
endfunction
t=0:10;
y0=[2 1];
sol=ode(y0,0,t,predprey);
count=0;
for i=1:2:22
    disp(count,"istep=")
    disp(count,"time=")
    disp(sol(i),"y1=")
```

```
    disp(sol(i+1),"y2=")
    disp("
      ------------------------------------------------------------
      ")
    count=count+1;
end
```

Chapter 29

Finite Difference Elliptic Equations

Scilab code Exa 29.1 Temperature of a heated plate with fixed boundary conditions

```
//clc()
wf = 1.5;
for i = 1:5
    for j = 1:5
        T(i,j) = 0;
        if j == 1 then
            T(i,j) = 0;//C
        else
            if j == 5 then
                 T(i,j) = 100;//C
            end
        end
        if i == 1 then
            T(i,j) = 75;//C
        else
            if i == 5 then
                T(i,j) = 50;//C
            end
        end
```

```
    end
end
e = 100;
while e>1
for i=1:5
    for j = 1:5
        if i>1 & j>1 & i<5 & j<5 then
        Tn(i,j) = (T(i + 1,j) + T(i-1,j) + T(i,j+1)
           + T(i,j-1))/4;
        Tn(i,j) = wf * Tn(i,j) + (1-wf)*T(i,j);
            if i==2 & j==2 then
            e = abs((Tn(i,j) - T(i,j)) * 100/ (Tn(i,
               j)));
            end
        T(i,j) = Tn(i,j);
        end
    end
end
end
disp(T(2:4,2:4),"for error < 1, the temperatures are
   ")
```

Scilab code Exa 29.2 Flux distribution for a heated plate

```
//clc()
k = 0.49;//cal/(s.cm.C)
T1 = 33.29755;//C
T2 = 75;//C
T3 = 63.21152;//C
T4 = 0;//C
qx = -k * (T1 - T2) / (2*10);
qy = -k * (T3 - T4) / (2*10);
disp(qx,"qx = ")
disp(qy,"qy = ")
q = (qx^2 + qy^2)^0.5;
```

```
t = (atan(qy/qx));
r = t * 180/(%pi);
disp(r,"Thus, the flux is direced at the angle")
```

Scilab code Exa 29.3 Heated plate with a insulated edge

```
//clc()
//since, insulation is done at one end,the general
   equation becomes(j=0),
//T(i+1,0) + T(i-1,0) + 2*T(i,1) -2*y*(dT/dy) - 4*T(
   i,0) = 0
//the derivative is zero and we get , T(i+1,0) + T(i
   -1,0) + 2*T(i,1) - 4*T(i,0) = 0
//the simultaneous equations obtained can be written
    in matrix form as follows,
A =
   [4,-1,0,-2,0,0,0,0,0,0,0,0;-1,4,-1,0,-2,0,0,0,0,0,0,0;0,-1,4,0,0,

disp(A,"A = ")
B = [75;0;50;75;0;50;75;0;50;175;100;150];
disp(B,"B = ")
T = inv(A)*B;
T10 = det(T(1,1));
T20 = det(T(2,1));
T30 = det(T(3,1));
T11 = det(T(4,1));
T21 = det(T(5,1));
T31 = det(T(6,1));
T12 = det(T(7,1));
T22 = det(T(8,1));
T32 = det(T(9,1));
T13 = det(T(10,1));
T23 = det(T(11,1));
T33 = det(T(12,1));
disp(T10,"T10 = ")
```

```
disp(T20,"T20 = ")
disp(T30,"T30 = ")
disp(T11,"T11 = ")
disp(T21,"T21 = ")
disp(T31,"T31 = ")
disp(T12,"T12 = ")
disp(T22,"T22 = ")
disp(T32,"T32 = ")
disp(T13,"T13 = ")
disp(T23,"T23 = ")
disp(T33,"T33 = ")
```

Scilab code Exa 29.4 Heated plate with an irregular boundary

```
//clc()
//for irregular boundary,
//x = y
alpha1 = 0.732;
beta1 = 0.732;
alpha2 = 1;
beta2 = 1;
//substituting the above value to obtain
   simultaneous equation we get the following matrix
   ,
A =
   [4,-0.845,0,-0.845,0,0,0,0,0;-1,4,-1,0,-1,0,0,0,0;0,-1,4,0,0,-1,0,
disp(A,"A = ")
B = [173.2;75;125;75;0;50;175;100;150];
disp(B,"B = ")
T = inv(A)*B;
T11 = det(T(1,1));
T21 = det(T(2,1));
T31 = det(T(3,1));
T12 = det(T(4,1));
```

```
T22 = det(T(5,1));
T32 = det(T(6,1));
T13 = det(T(7,1));
T23 = det(T(8,1));
T33 = det(T(9,1));
disp(T11,"T11 = ")
disp(T21,"T21 = ")
disp(T31,"T31 = ")
disp(T12,"T12 = ")
disp(T22,"T22 = ")
disp(T32,"T32 = ")
disp(T13,"T13 = ")
disp(T23,"T23 = ")
disp(T33,"T33 = ")
```

Chapter 30

Finite Difference Parabolic Equations

Scilab code Exa 30.1 Explicit solution of a one dimensional heat conduction equati

```
//clc()
l = 10;//cm
k1 = 0.49;//cal/(s.cm.C)
x = 2;//cm
dt = 0.1;//sec
T0 = 100;//C
T10 = 50;//C
C = 0.2174;//cal/(g.C)
rho = 2.7;//g/cm^3
k = k1/(C*rho);
L = k * dt / x^2;
disp(L,"L =")
T(1,1) = 100;
T(1,2) = 0;
T(1,3) = 0;
T(1,4) = 0;
T(1,5) = 0;
T(1,6) = 50;
T(2,1) = 100;
```

```
T(2,6) = 50;
for i = 2:3
    for j = 2:5
        T(i,j) = T(i-1,j) + L * (T(i-1,j+1) - 2* T(i-1,j) + T(i-1,j-1));
    end
end
disp(T(2,2),"T11 = ")
disp(T(2,3),"T12 = ")
disp(T(2,4),"T13 = ")
disp(T(2,5),"T14 = ")
disp(T(3,2),"T21 = ")
disp(T(3,3),"T22 = ")
disp(T(3,4),"T23 = ")
disp(T(3,5),"T24 = ")
```

Scilab code Exa 30.2 Simple implicit solution of a heat conduction equation

```
//clc()
l = 10;//cm
k1 = 0.49;//cal/(s.cm.C)
x = 2;//cm
dt = 0.1;//sec
C = 0.2174;//cal/(g.C)
rho = 2.7;//g/cm^3
k = k1/(C*rho);
L = k * dt / x^2;
disp(L,"L =")
//now, at t = 0, 1.04175 *T'1 + 0.020875 *T'2 = 0 + 0.020875*100
//similarly getting other simultaneous eqautions,we get the following matrix
A = [1.04175,-0.020875,0,0;-0.020875,1.04175,-0.020875,0;0,-0.020875,
```

```
B = [2.0875;0;0;1.04375];
X = inv(A) * B;
T11 = det(X(1,1));
T12 = det(X(2,1));
T13 = det(X(3,1));
T14 = det(X(4,1));
disp("for t = 0")
disp(T11,"T11 = ")
disp(T12,"T12 = ")
disp(T13,"T13 = ")
disp(T14,"T14 = ")
//to solve for t = 0.2, the right hand side vector is modified to D,
D = [4.09215;0.04059;0.02090;2.04069];
Y = inv(A)*D;
T21 = det(Y(1,1));
T22 = det(Y(2,1));
T23 = det(Y(3,1));
T24 = det(Y(4,1));
disp("for t = 0.2")
disp(T21,"T21 = ")
disp(T22,"T22 = ")
disp(T23,"T23 = ")
disp(T24,"T24 = ")
```

Scilab code Exa 30.3 `Crank Nicolson solution to the heat conduction equation`

```
//clc()
//using crank nicolson method, we get simultaneous equations which can be simplified to following matrics
A =
   [2.04175,-0.020875,0,0;-0.020875,2.04175,-0.020875,0;0,-0.020875,

B = [4.175;0;0;2.0875];
```

```
X = inv(A)*B;
disp("at t = 0.1 s")
disp(det(X(1,1)),"T11 = ")
disp(det(X(2,1)),"T12 = ")
disp(det(X(3,1)),"T13 = ")
disp(det(X(4,1)),"T14 = ")
C = [8.1801;0.0841;0.0427;4.0901];
Y = inv(A)*C;
disp("at t = 0.2 s")
disp(det(Y(1,1)),"T21 = ")
disp(det(Y(2,1)),"T22 = ")
disp(det(Y(3,1)),"T23 = ")
disp(det(Y(4,1)),"T24 = ")
```

Scilab code Exa 30.4 Comparison of Analytical and Numerical solution

```
//clc()
x = 2;//cm
L = 10;//cm
k = 0.835;//cm^2/s
t = 10;//s
n = 1:10000;
T = 324.* (x/L + sum(2.*((-1)^n).*sin(n.*x/L).*exp(-
    n^2.* %pi^2.* k.* t / L^2)/(n.*%pi)));
disp(T,"T[2,10]analytical = ")
Tex = 64.97;
disp(Tex,"T[2,10]explicit = ")
Tim = 64.49;
disp(Tim,"T[2,10]implicit = ")
Tcn = 64.73;
disp(Tcn,"T[2,10]crank-nicolson = ")
```

Scilab code Exa 30.5 ADI Method

```
//clc()
x = 10;//cm
L = 0.0835;
t1 = 5;
//for first step t = 5 is applied to nodes (1,1) ,
    (1,2) and (1,3) to yield following matrices
A =
    [2.167,-0.0835,0;-0.0835,2.167,-0.0835;0,-0.0835,2.167];

B = [6.2625;6.2625;14.6125];
X = inv(A)*B;
disp("At t = 5 s")
disp(det(X(1,1)),"T11 = ")
disp(det(X(2,1)),"T12 = ")
disp(det(X(3,1)),"T13 = ")
//similarly we get,
T21 = 0.1274;
T22 = 0.2900;
T23 = 4.1291;
T31 = 2.0181;
T32 = 2.2477;
T33 = 6.0256;
disp(T21,"T21 = ")
disp(T22,"T22 = ")
disp(T23,"T23 = ")
disp(T31,"T31 = ")
disp(T32,"T32 = ")
disp(T33,"T33 = ")
C = [13.0639;0.2577;8.0619];
Y = inv(A)*C;
disp("At t = 10 s")
disp(det(Y(1,1)),"T11 = ")
disp(det(Y(2,1)),"T12 = ")
disp(det(Y(3,1)),"T13 = ")
//similarly we get,
T21 = 6.1683;
T22 = 0.8238;
T23 = 4.2359;
```

```
T31 = 13.1120;
T32 = 8.3207;
T33 = 11.3606;
disp(T21,"T21 = ")
disp(T22,"T22 = ")
disp(T23,"T23 = ")
disp(T31,"T31 = ")
disp(T32,"T32 = ")
disp(T33,"T33 = ")
```

Chapter 31

Finite Element Method

Scilab code Exa 31.1 Analytical Solution for Heated Rod

```
clc;
clear;
//d2T/dx2=-10; equation to be solved
//T(0,t)=40; boundary condition
//T(10,t)=200; boundary condition
//f(x)=10; uniform heat source
//we assume a solution T=a*X^2 + b*x +c
//differentiating twice we get d2T/dx2=2*a
a=-10/2;
//using first boundary condition
c=40;
//using second boundary condtion
b=66;
//hence final solution T=-5*x^2 + 66*x + 40
function T=f(x)
    T=-5*x^2 + 66*x + 40
endfunction
count=1;
for i=0:0.1:11
```

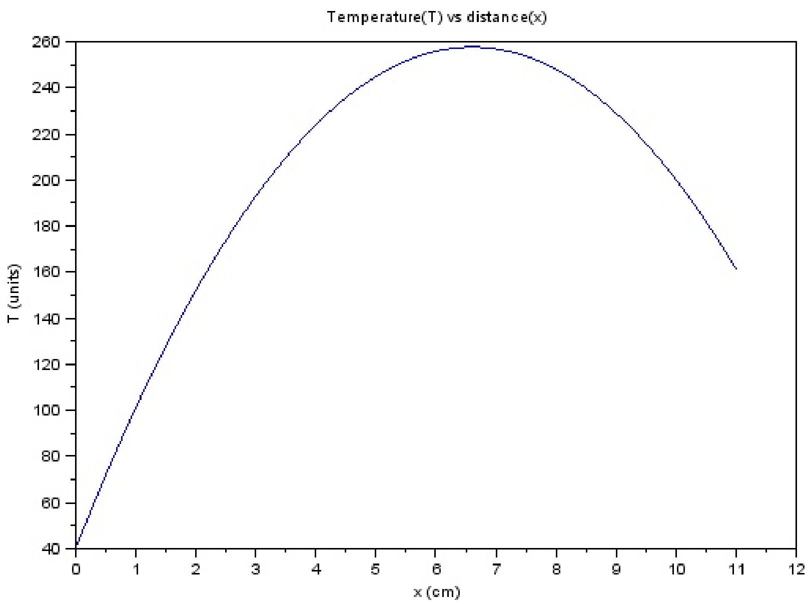

Figure 31.1: Analytical Solution for Heated Rod

```
    T(count)=f(i);
    count=count+1;
end
x=0:0.1:11
plot(x,T)
xtitle("Temperature(T) vs distance(x)","x (cm)","T (
   units)")
```

Scilab code Exa 31.2 `Element Equation for Heated Rod`

```
clc;
clear;
xf=10;//cm
xe=2.5;//cm
//T(0,t)=40; boundary condition
//T(10,t)=200; boundary condition
//f(x)=10; uniform heat source
function y=f(x)
    y=10*(xe-x)/xe;
endfunction
int1=intg(0,xe,f)
function y=g(x)
    y=10*(x-0)/xe;
endfunction
int2=intg(0,xe,g)
disp("The results are:")
disp("0.4*T1-0.4*T2=-(dT/dx)*x1 + c1")
disp(int1,"where c1=")
disp("and")
disp("-0.4*T1+0.4*T2=-(dT/dx)*x2 + c2")
disp(int2,"where c2=")
```

Scilab code Exa 31.3 `Temperature of a heated plate`

```
//clc()
wf = 01.5;
for i = 1:41
    for j = 1:41
        T(i,j) = 0;
        if j == 1 then
            T(i,j) = 0;//C
        else
            if j == 41 then
                T(i,j) = 100;//C
            end
        end
        if i == 1 then
            T(i,j) = 75;//C
        else
            if i == 41 then
                T(i,j) = 50;//C
            end
        end
    end
end
e = 100;
while e>1
for i=1:41
    for j = 1:41
        if i>1 & j>1 & i<41 & j<41 then
        Tn(i,j) = (T(i + 1,j) + T(i-1,j) + T(i,j+1)
           + T(i,j-1))/4;
        Tn(i,j) = wf * Tn(i,j) + (1-wf)*T(i,j);
            if i==2 & j==2 then
            e = abs((Tn(i,j) - T(i,j)) * 100/ (Tn(i,
               j)));
            end
        T(i,j) = Tn(i,j);
        end
    end
end
end
```

```
disp(T,"for error < 1, the temperatures are")
```