Practical 1

(Iterative Calculation)

(a) Program for iterative Calculation.

Write the scilab code for the following consider the following:

e^x = 1 + x + x^2 /2! + x^3 /3!+......+x^n/n!

Evaluate $e^{0.3}$ and compare with the true value 1.3498588. Use six terms to evaluate each series and compute true and approximate relative errors(upto three significant figure) as terms are added.

```
n=3;//number of significant figures

es=0.3*(10^(2-n));//percent, spcified error criterion

x=0.3;

f(1)=1;//first estimate f=e^x=1

ft=1.3498588;//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
```

OUTPUT:

term number=

1.

Result=

1.

True % relative error=

25.918178

Approximate estimate of error(%)=

100.

---

term number=

2.

Result=

1.3

True % relative error=

3.6936308

Approximate estimate of error(%)=

23.076923

---

term number=

3.

Result=

1.345

True % relative error=

0.3599488

Approximate estimate of error(%)=

3.3457249

----

term number=

4.

Result=

1.3495

True % relative error=

0.0265806

Approximate estimate of error(%)=

0.3334568

----

term number=

5.

Result=

1.3498375

True % relative error=

0.0015779

Approximate estimate of error(%)=

0.0250030

(b) Program to calculate the roots of quadratic equation using the formula.
Write scilab code to evaluate roots of quadratic equation x^2+6x+5=0.

```
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
```

```
Scilab 5.5.2 Console
File  Edit  Control  Applications  ?
Scilab 5.5.2 Console

-->exec('C:\Users\Admin\Desktop\Vishal\NSM\Quadraticeqn.sce', -1)
Input the value of a:1
Input the value of b:6
Input the value of c:5

 The roots are:

  - 1.

 and

  - 5.

-->
```

Practical No.2

(Solution of algebraic and transcendental equations)

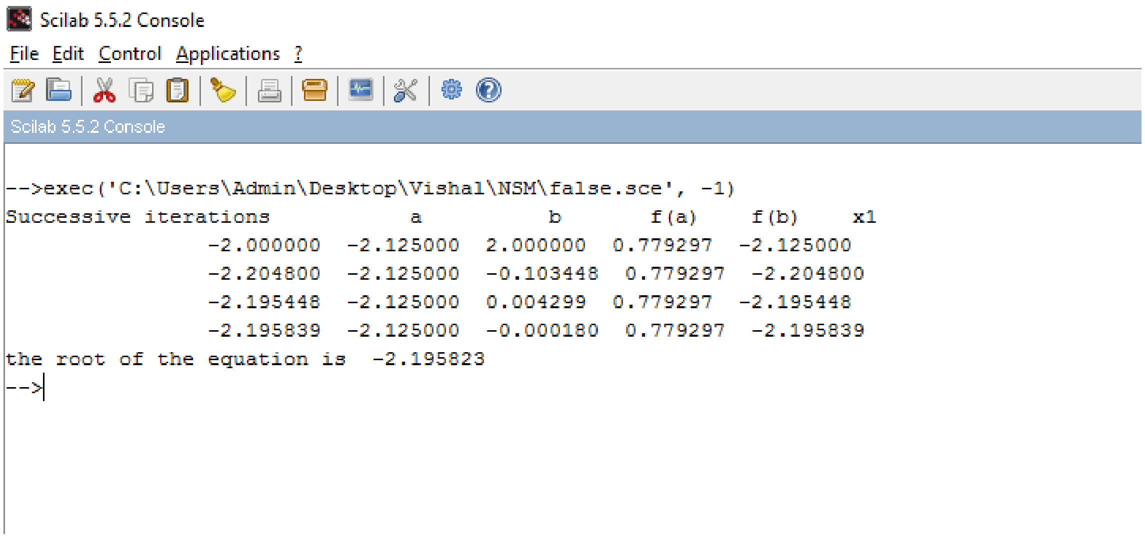
Write a scilab code to find the real root of the equation x^3-x-4 = 0 using bisection method correct to four places of decimal

```
deff('y=f(x)','y=x^3-x-4');
x1=1,x2=2;//f(0) is negative and f(1) is positive
d=0.0001;//for accuracy of root
c=1;
printf('Successive approximations \t  x1\t  \t\x2\t  \tm\t  \tf(m)\n');
while abs(x1-x2)>d
   m=(x1+x2)/2;
   printf('                    \t%f\t%f\t%f\t%f\n',x1,x2,m,f(m));
   if f(m)*f(x1)>0
      x1=m;
   else
      x2=m;
   end
   c=c+1;//to count number of iterations
end
printf('the solution of equation after %i iteration is %0.4f',c,m);
```

```
-----------------------------------------------------

-->exec('C:\Users\Admin\Desktop\Vishal\NSM\Bisection.sce', -1)
Successive approximations        x1            x2            m             f(m)
                              1.000000      2.000000      1.500000      -2.125000
                              1.500000      2.000000      1.750000      -0.390625
                              1.750000      2.000000      1.875000      0.716797
                              1.750000      1.875000      1.812500      0.141846
                              1.750000      1.812500      1.781250      -0.129608
                              1.781250      1.812500      1.796875      0.004803
                              1.781250      1.796875      1.789063      -0.062730
                              1.789063      1.796875      1.792969      -0.029046
                              1.792969      1.796875      1.794922      -0.012142
                              1.794922      1.796875      1.795898      -0.003675
                              1.795898      1.796875      1.796387      0.000563
                              1.795898      1.796387      1.796143      -0.001556
                              1.796143      1.796387      1.796265      -0.000497
                              1.796265      1.796387      1.796326      0.000033
the solution of equation after 15 iteration is 1.796326
-->
```

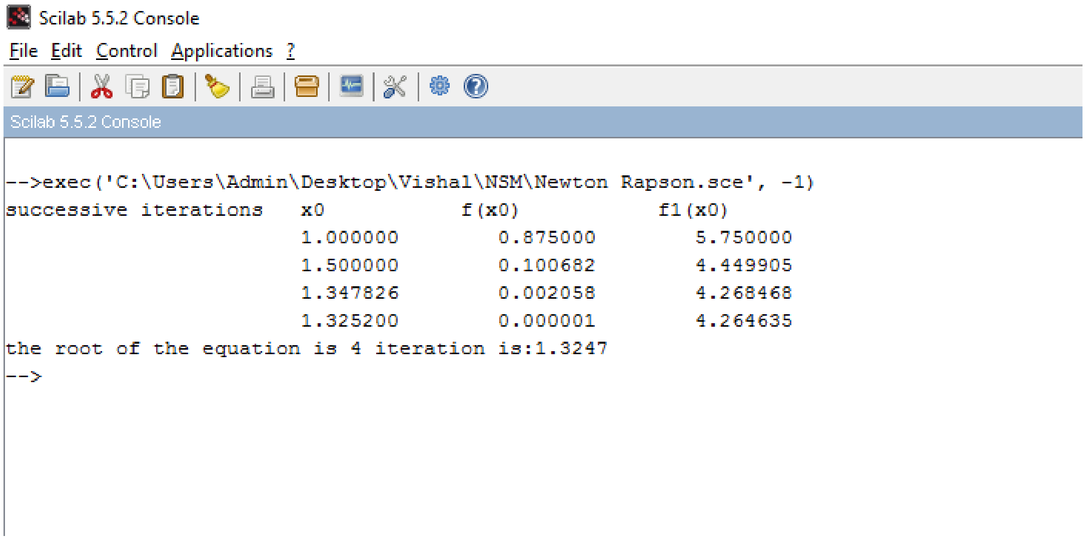
Write a scilab code to find the real root of the equation x^3-x-4 = 0 using False Position method correct to four places of decimal

```
deff('y=f(x)','y=x^3-3*x+4');
a=-2,b=-3;//f(1) is negative and f(2) is positive
d=0.00001;
printf('Successive iterations   \ta\t   b\t   f(a)\t   f(b)\t   x1\n');
for i=1:25
   x1=(a*f(b)-b*f(a))/(f(b)-f(a));
   if(f(a)*f(x1))>0
      b=x1;
   else
      a=x1;
end
if abs(f(x1))<d then
   break
end
printf('           \t%f  %f  %f  %f  %f\n',a,b,f(a),f(b),x1);
end
printf('the root of the equation is  %f',x1);
```



Write a scilab code to find the real root of the equation x^3-x-1= 0 using Newton Raphson method by taking initial root 1 .

```
deff('y=f(x)','y=x^3-x-1');
deff('y1=f1(x)','y1=3*x^2-1');
x0=1;//initial value
d=0.00001;
c=0,n=1;
printf('successive iterations \tx0\t     f(x0)\t     f1(x0)\n');
while n==1
   x2=x0;
   x1=x0-(f(x0)/f1(x0));
   x0=x1;
   printf('                  \t%f\t%f\t%f\n',x2,f(x1),f1(x1));
   c=c+1;
   if abs(f(x0))<d then
      break;
      end
end
printf('the root of the equation is %i iteration is:%0.4f',c,x0);
```



Practical No.3

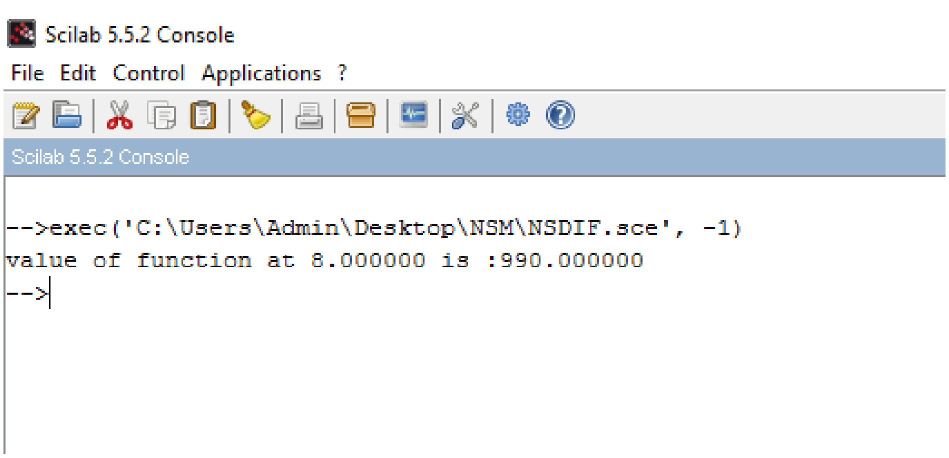
(Interpolation)

Write a scilab code to find f(8) using Newton's Forward Difference interpolation formula for the following data.

x	**1**	**3**	**5**	**7**
F(X)	**24**	**120**	**336**	**720**

```
x=[1 3 5 7];

y=[24 120 336 720];

h=2 //interval between values of x

c=1;

for i=1:3

   d1(c)=y(i+1)-y(i);

   c=c+1;

end

c=1;

for i=1:2

   d2(c)=d1(i+1)-d1(i);
```

```
   c=c+1;
end
c=1;
for i=1:1
   d3(c)=d2(i+1)-d2(i);
   c=c+1;
end
d=[d1(1) d2(1) d3(1)];
x0=8; // value at 8;
y_x=y(1);
p=(x0-1)/2;
for i=1:3
   pp=1;
   for j=1:i
      pp=pp*(p-(j-1))
end
y_x=y_x+(pp*d(i))/factorial(i);
end
printf('value of function at %f is :%f',x0,y_x);
```

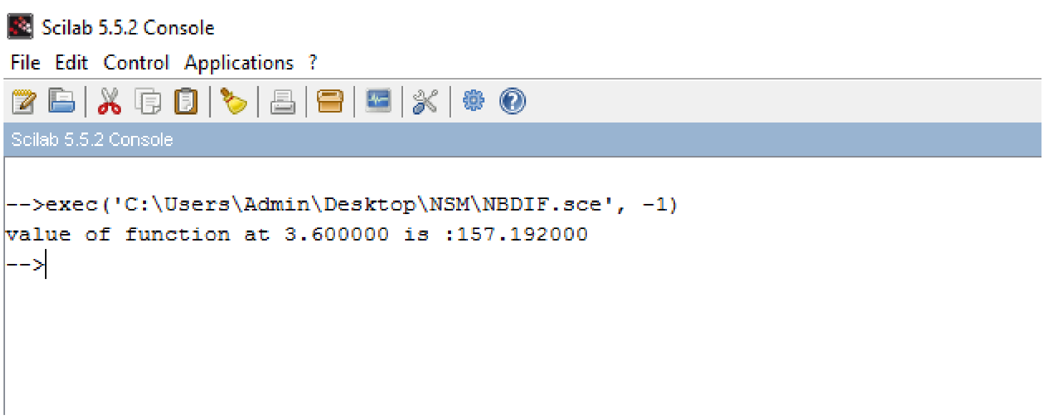


Write a scilab code to find f(8) using Newton's Backward Difference interpolation formula for the following data.

X	0	1	2	3	4
F(x)	7	17	45	103	203

```
x=[0 1 2 3 4];
y=[7 17 45 103 203];
h=1;
c=1;
for i=1:4
   d1(c)=y(i+1)-y(i);
   c=c+1;
end
c=1;
for i=1:3
   d2(c)=d1(i+1)-d1(i);
   c=c+1;
end
c=1;
for i=1:2
   d3(c)=d2(i+1)-d2(i);
   c=c+1;
end
c=1;
for i=1:1
   d4(c)=d3(i+1)-d3(i);
```

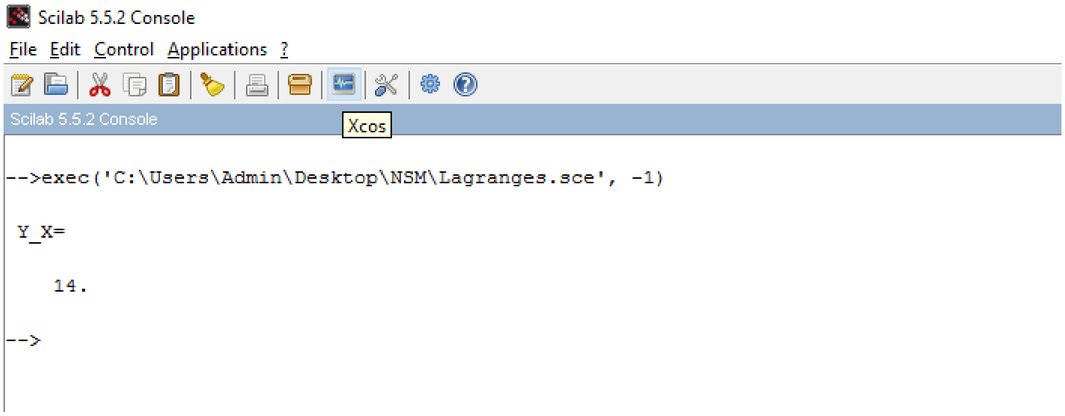
```
    c=c+1;
end
c=1;
d=[d1(4) d2(3) d3(2) d4(1)];
x0=3.6;
pp=1;
y_x=y(5);
p=(x0-x(5))/h;
for i=1:4
    pp=1;
    for j=1:i
        pp=pp*(p+(j-1))
end
y_x=y_x+((pp*d(i))/factorial(i));
end
printf('value of function at %f is :%f',x0,y_x);
```



Use Lagranges interpolation formula to find the values of y when x=8 from the following table

X	5	6	9
y	12	13	14

```
// Lagrange's interpolation formula
y=[5 6 9];
x=[12 13 14];
y_x=8;
Y_X=0;
poly(0,'y');
for i=1:3
  p=x(i);
  for j=1:3
    if i~=j
      p=p*((y_x-y(j))/(y(i)-y(j)))
end
end
Y_X=Y_X+p;
end
disp(Y_X,'Y_X=');
```



Practical No.4

(Solving linear system of equations by iterative methods)

Write the scilab code for the following.

Use the Guass Jordan method to solve the following system:

$X_1+x_2+x_3=90$

$2x_1+3x_2+6x_3=370$

$3x_1-8x_2-4x_3=-340$

```
A=[1,1,1,90;2,3,6,370;3,-8,-4,-340];//Augmented matrix
for i=1:3
    j=i
    while(A(i,i)==0&j<=3)
for k=1:4
    B(1,k)=A(j+1,k)
    A(j+1,k)=A(i,k)
    A(i,k)=B(1,k)
end
disp(A);
```

```
j=j+1;
end
disp(A);
for k=4: -1: i
     A(i,k)=A(i,k)/A(i,i)
end
disp(A)
    for k=1:3
       if(k~=i)
    l=A(k,i)/A(i,i)
    for m=i:4
           A(k,m)=A(k,m)-l*A(i,m)
    end
  end
end
disp(A)
end
for i=1:3
printf('\nx(%i)=%g\n',i,A(i,4))
           end
```

OUTPUT:

```
1.    1.    1.    90.
   2.    3.    6.    370.
   3.  - 8.  - 4.  - 340.

   1.    1.    1.    90.
   2.    3.    6.    370.
   3.  - 8.  - 4.  - 340.
```

1.	1.	1.	90.
0.	1.	4.	190.
0.	- 11.	- 7.	- 610.

1.	1.	1.	90.
0.	1.	4.	190.
0.	- 11.	- 7.	- 610.

1.	1.	1.	90.
0.	1.	4.	190.
0.	- 11.	- 7.	- 610.

1.	0.	- 3.	- 100.
0.	1.	4.	190.
0.	0.	37.	1480.

1.	0.	- 3.	- 100.
0.	1.	4.	190.
0.	0.	37.	1480.

1.	0.	- 3.	- 100.
0.	1.	4.	190.
0.	0.	1.	40.

1.	0.	0.	20.
0.	1.	0.	30.
0.	0.	1.	40.

x(1)=20

x(2)=30

x(3)=40

Practical No.5

(Numerical Integration)

Sove the given integral using Trapezoidal Method

```
function [i]=trapezoidal(a, b, n, f)
 h=(b-a)/n
x=(a:h:b)
y=f(x)
m=length(y)
i=y(1)+y(m)
 for j=2:m-1
  i=i+2*y(j)
  end
  i=h*i/2
   return(i)
   endfunction
```

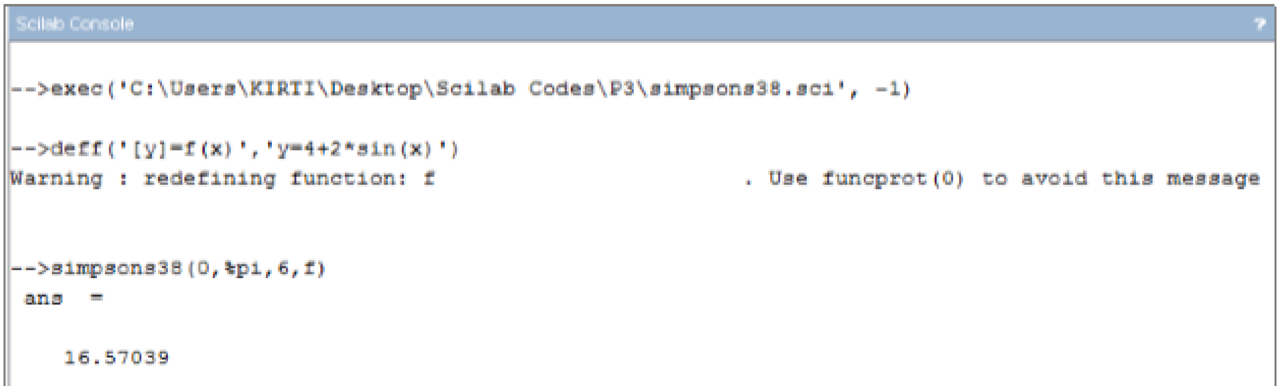
Scilab Console

```
-->deff('[y]=f(x)','y=(1+x^2)^(-1)')

-->trapezoidal(0,1,4,f)
 ans  =

    0.7827941

-->deff('[y]=f(x)','y=4+2*sin(x)')
Warning : redefining function: f


-->trapezoidal(0,%pi,6,f)
 ans  =

    16.474565
```

Simpson's 3/8th rule

```
function [i]=simpsons38(a, b, n, f)
  h=(b-a)/n
  x=(a:h:b)
  y=f(x)
  m=length(y)
  i=y(1)+y(m)
  for j=2:m-1
    v=modulo(j-1,3);
    if v==0 then
      i=i+2*y(j)
    else
      i=i+3*y(j)
    end
  end
  i=3*h*i/8
```

```
return(i)
endfunction
```



```
Scilab Console

-->exec('C:\Users\KIRTI\Desktop\Scilab Codes\P3\simpsons38.sci', -1)

-->deff('[y]=f(x)','y=4+2*sin(x)')
Warning : redefining function: f                    . Use funcprot(0) to avoid this message


-->simpsons38(0,%pi,6,f)
 ans  =

    16.57039
```

Simpsons 1/3 rd rule

```
function [i]=simpsons13(a, b, n, f)
  h=(b-a)/n
  x=(a:h:b)
  y=f(x)
  m=length(y)
  i=y(1)+y(m)
  for j=2:m-1
     if modulo(j,2)==0 then
        i=i+4*y(j)
     else
        i=i+2*y(j)
     end
   end
```

```
   i=h*i/3
return(i)
endfunction
```

```
Scilab Console

-->deff('[y]=f(x)','y=1+x^4')
Warning : redefining function: f

-->simpsons13(0,2,4,f)
 ans  =

    8.4166667
```

Practical No.6

(Solution of Differential equations)

Solve the d.e. to find y(0.2) using euler's method

dy/dx=x^2+2y,y(0)=1

```
function[Y0]=eular(X0,Y0,h,yest,f)
   //X0,Y0,h have usual meaning and yest is value for which Y is to be obtained.
   n=(yest-X0)/h
   for i=1:n
      Y0=Y0+f(X0,Y0)*h;
```

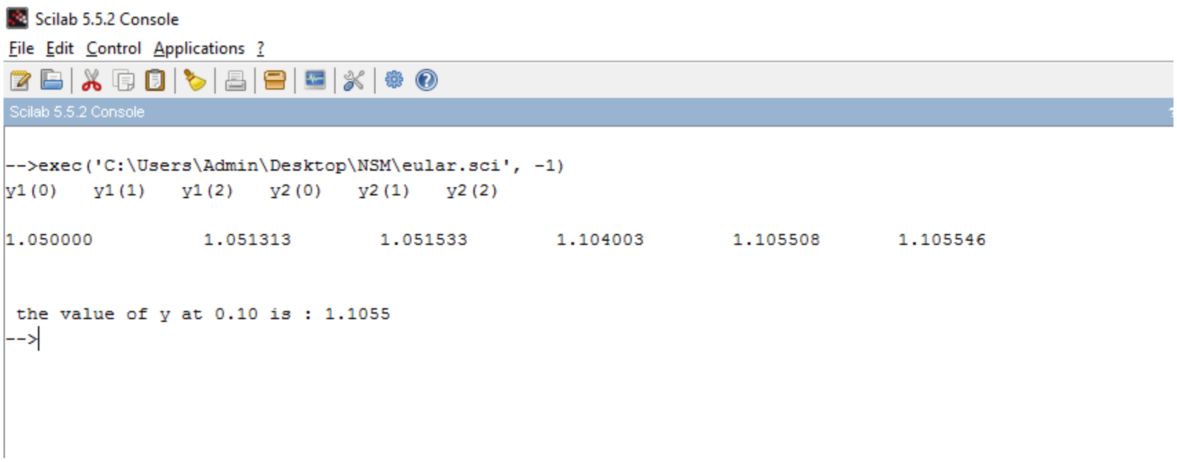
```
    X0=X0+h;
    disp(Y0)
  end
endfunction

deff('[y]=f(a,b)','y=a^2+2*b')
eular(0,1,0.2,0.2,f)
```

```
-->exec('C:\Users\Admin\Desktop\NSM\eular.sci', -1)
Warning : redefining function: f                       . Use funcprot(0) to avoid this message

    1.4
```

Determine the value of y when x=0.1 using modified Euler's method given that y(0) = 1 and x^2 + y take h = 0.05

```
//Euler's modified method
h=0.05
f=1;
deff('z=f1(x,y)','z=x^2+y');
x=0:0.05:0.1
y1=0;
y1(1)=f+h*f1(x(1),f);
y1(2)=f+h*(f1(x(1),f)+f1(x(2),y1(1)))/2;
y1(3)=f+h*(f1(x(1),f)+f1(x(3),y1(2)))/2;
y2(1)=y1(2)+h*f1(x(2),y1(2));
y2(2)=y1(2)+h*(f1(x(2),y1(2))+f1(x(3),y2(1)))/2;
```

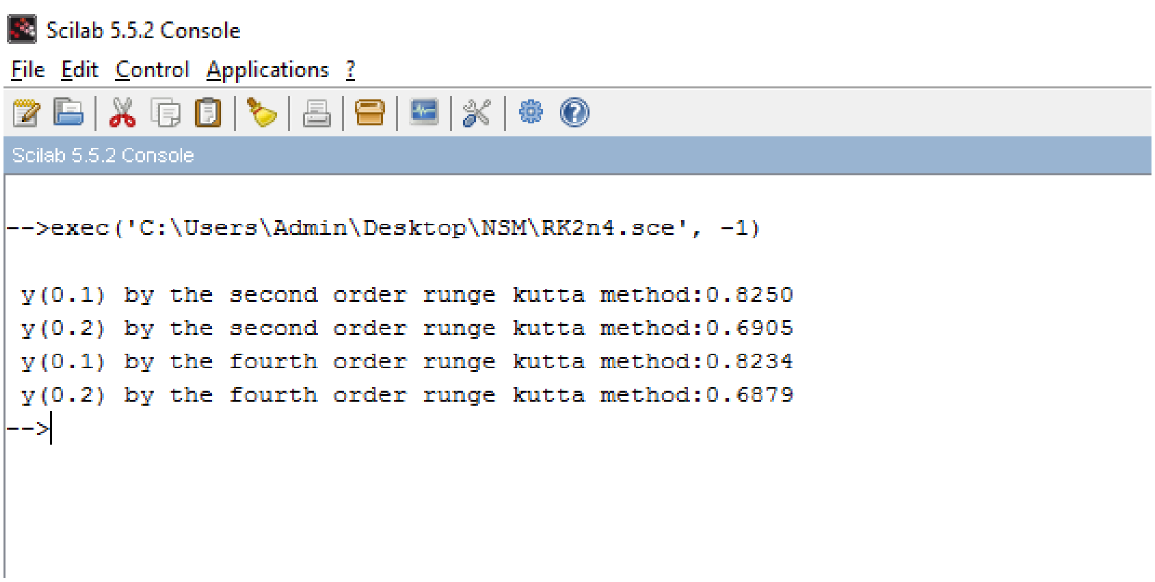
```
y2(3)=y1(2)+h*(f1(x(2),y1(2))+f1(x(3),y2(2)))/2;

printf('y1(0)\ty1(1)\ty1(2)\ty2(0)\ty2(1)\ty2(2)\n\n');

printf('%f\t  %f\t  %f\t  %f\t  %f\t %f\n',y1(1),y1(2),y1(3),y2(1),y2(2),y2(3));

printf('\n\n the value of y at %0.2f is : %0.4f',x(3),y2(3));
```



Determine the value of y(0.1) and y(0.2) using RK 2[nd] and 4th order methods given that y(0) = 1 and y' = x-2y take h=0.1

```
//Rung-kutta formula

deff('y=f(x,y)','y=x-2*y');

y=1,x=0,h=0.1;

k1=h*f(x,y);

k2=h*f(x+h,y+k1);

y1=y+(k1+k2)/2;

printf('\n y(0.1) by the second order runge kutta method:%0.4f',y1);

y=y1,x=0.1,h=0.1;

k1=h*f(x,y);

k2=h*f(x+h,y+k1);

y1=y+(k1+k2)/2;
```

```
printf('\n y(0.2) by the second order runge kutta method:%0.4f',y1);
y=1,x=0,h=0.1;
k1=h*f(x,y);
k2=h*f(x+h/2,y+k1/2);
k3=h*f(x+h/2,y+k2/2);
k4=h*f(x+h,y+k3);
y1=y+(k1+2*k2+2*k3+k4)/6;
printf('\n y(0.1) by the fourth order runge kutta method:%0.4f',y1);
y=y1,x=0.1,h=0.1;
k1=h*f(x,y);
k2=h*f(x+h/2,y+k1/2);
k3=h*f(x+h/2,y+k2/2);
k4=h*f(x+h,y+k3);
y1=y+(k1+2*k2+2*k3+k4)/6;
printf('\n y(0.2) by the fourth order runge kutta method:%0.4f',y1);
```



Practical No.7

(Regression)

Write and execute scilab code for the following:

Fit a straight line for the following data

X	1	2	3	4	5	6	7
y	0.5	2.5	2	4	3.5	6	5.5

```
//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")

```
-->exec('C:\Users\Admin\Documents\sl.sce', -1)

 sum of product of x and y =

    94.5

 sum of square of x=

    140.

 sum of all the x=

    28.

 sum of all the y=

    21.5

 a1=

    0.3035714

 a0=

    1.8571429

 The equation of the line obtained is y = a0+a1*x

-->
```

Practical No.8

Random Number Generation from Binomial Distribution

```
function [x]=binrand(N, n, pr)

for i=1:N

p=rand();

q=1-p;

x(i)=round(cdfbin("S",n,pr,1-pr,p,q));

end;

return(x)

endfunction
```

```
-->exec('C:\Users\Admin\Desktop\Distributions\binrand.sci', -1)
Warning : redefining function: binrand                 . Use funcprot(0) to avo


-->binrand(10,7,0.8)
 ans  =

    6.
    5.
    5.
    6.
    3.
    6.
    6.
    5.
    5.
    6.
```

Random Number Generation from Poisson distribution

```
function [x]=poissonrand(N, m)
for i=1:N
p=rand();
q=1-p;
x(i)=round(cdfpoi("S",m,p,q));
end;
return(x);
endfunction
```

```
-->exec('C:\Users\Admin\Desktop\Distributions\poissonrand.sci', -1)

-->poissonrand(10,6)
 ans  =

    7.
    2.
    6.
    3.
    5.
    8.
    6.
    5.
    4.
    8.
```

Random Number Generation from Uniform Distribution

```
function [x]=uniformrand(N, m, n)
   for i=1:N
      u=rand();
      x(i)=(n-m)*u+m;
   end
   return(x);
endfunction
```

```
-->exec('C:\Users\Admin\Desktop\Distributions\uniformrand.sci', -1)

-->uniformrand(20,3,5)
 ans  =

    3.2411992
    3.5710728
    4.7215029
    4.6988203
    4.0514122
    4.986242
    4.2977126
    4.9846382
    3.100084
    4.4971013
    3.8208118
    4.2169053
    4.7088422
    3.1285293
    4.6558166
    4.8524688
    4.1334423
    4.1423278
    4.6320221
    3.1137856
```