1.2) Roots Of Quadratic

```
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
```

Output:

Input the value of a :

```
5
Input the value of b :
4
Input the value of c :
3

 The roots are:

  -0.4 + 0.663325i

  -0.4 - 0.663325i
```

1.3) Infinite Series

```
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:")
   i=i+1;
   term=term*x/i;
   test=sum;
   sum=sum+term;
end
disp(f(x),"Exact Value")
```

output:
Input value of x:
5

i:

0.

term:

1.

sum:

1.

i:

1.

2.3) Secant method

```
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=")
```

output:

x=

0.
1.
0.6126998
0.5638384
0.5671704

et=

-8.0326344

3.2) Newton backward interpolation

```
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);
```

output:
value of function at 3.600000 is :157.192000

3.3) lagrange's interpolation

```
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=');
```

output:

Y_X=

14.

4.1) gauss Jordan

```
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=")
```

output:

Equation in matrix form can be written as

```
  3.   -0.1  -0.2   7.85
  0.1   7.   -0.3  -19.3
  0.3  -0.2   10.   71.4
```

final matrix=

```
  1.   0.   0.   3.
  0.   1.   0.  -2.5
```

0. 0. 1. 7.

x1=

3.

x2=

-2.5

x3=

7.

6.1) Trapezoidal Rule

```
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
disp(i)
endfunction
deff('[y]=f(x)','y=(1+x^2)^(-1)')
trapezoidal(0,1,4,f)
```

output:

```
0.7827941
```

8.1) Linear Regression

```
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")
```

output:

```
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

9.2) Fit Binomial Distribution

```
function [expfre]=binfit(x, obsfre)
   n=length(x)-1;
   N=sum(obsfre);
   xbar=sum(x*obsfre')/N;
   p=xbar/n;
   q=1-p;
   for r=0:n
      prob(r+1)=factorial(n)*p^r*q^(n-
r)/(factorial(r)*factorial(n-r));
   end
   expfre=round(prob*N);
   printf('Expected frequencies\n');
   printf('.....\n')

endfunction
x=[0,1,2,3,4,5];
obsfre=[12,25,36,21,10,8];
binfit(x,obsfre)
```

output:

Expected frequencies

.....

ans =

7.

26.

38.

29.

11.

2.

10.1) Uniform Distribution

```
function [expfre]=unifit(x, obsfre)
   n=length(x)-1;
   N=sum(obsfre);
   xbar=sum(x*obsfre')/N;
   prob(1)=1/(n+1);
   for r=0:n
      prob(r+1)=1/(n+1);
   end
   expfre=round(prob*N);
   printf('Expected frequencies\n');
   printf('.....\n')
   endfunction
x=[0,1,2,3,4,5];
obsfre=[12,50,100,80,45,5];
unifit(x,obsfre)
```

output:

Expected frequencies

.....

ans =

49.

49.

49.

49.

49.

49.