1 Bisection Method

```
# A python programm for bisection method
import math
import numpy
def f(x):
        function = (x*x)-3
        return function
print'*****************************'
print 'THE BISECTION METHOD'
print'*****************************'
x=float(input('Enter the initialvalue '))
y=float(input('Enter the finalvalue '))
n=int(input('Enter the number of  iterations '))
i=0
while(i<n):
        c=(x+y)/2.0
        if f(c)==0:# or (y-x)/2.0 < t:
                print c,'is the root'
        else:
                        if f(x)*f(c)<0:
                                y=c
                        else:
                                x=c
        i += 1
print c, 'is the root'
```

2 Eigen Value Method

```
#EIGEN VALUE METHOD
import numpy
import scipy as sp
import scipy.linalg as la
print'********************'
print 'EIGEN VALUE PROBLEM'
print'********************'
n=input('Enter the order of the matrix ')
print'********************'
print('The matrix is ')
A=sp.zeros((n,n)) #matrix creation for storing the values
for i in range(0,n):
        for j in range(0,n):
                print'A[',i,'][',j,'] :'
                A[i][j]=input()
w,v=la. eig(A)
print 'Eigen values are', w
```

3 Euler Method

```
# A python programm for Euler method for solving Differential equation 3x^2+1
print'EULER METHOD FOR 3X^2+1'
from numpy import *
from math import ceil
import math
def f(x,y):
        return(3*x**2+1)
print'**************************'
print 'EULER METHOD OF SOLVING DIFFERENTIAL EQUATIONS'
print'**************************'
x=float(input('Enter the initial conditions for x '))
y=float(input('Enter the initial conditions for y '))
print'**************************'
z=float(input('Enter the final value of the independent variable '))
print'**************************'
h=float(input('Enter the Enter the step length '))
print'**************************'
c=math.ceil((z-x)/h)
n=int(c)
for i in range(0,n):
        y=y+(h*f(x+i*h,y))
print 'The solution is y(',z,')= ',y
```

4 Gauss Elimination Method

```
#Gauss Elimination method
import numpy
import math
from numpy import *
import numpy as np
from math import fabs
print'********************'
print 'GUASS ELIMINATION METHOD'
print'********************'
n=int(input('The number of equations '))
print'********************'
print'********************'
print'Enter the LHS and RHS of the system row wise'
print'********************'
m=n+1
a = [[0] * m for i in range(n+1)]
x=(zeros(n))
for i in range(0,n):
        print'Enter the coeficients of equation number ', i
        for j in range(0,m):
                z=float(input('values '))
                a[i][j]=z
print'********************'
print'The given matrix is '
        for i in range(0,n):
                for j in range(0,m):
                        print '\t', a[i][j],
                print '\n'
print'********************'
for l in range(0,n):
        if(a[l][l]==0):
                for k in range(l+1, n):
                        if(a[k][l]!=0):
                                break
                                for j in range(0,n+1):
                                        s=a[l][j]
                                        a[l][j]=a[k][j]
                                        a[k][j]=s
        t=a[l][l]
        for j in range(0,n+1):
                a[l][j]=a[l][j]/t
        for i in range(l+1,n):
                s=a[i][l]
                for j in range(0,n+1):
```

```
                        a[i][j]=a[i][j]-s*a[l][j]
print'********************'
print'The equivalent matrix of the given matrix is '
for i in range(0,n):
        for j in range(0,n+1):
                print '\t',a[i][j],
        print '\n'
print'********************'
x[n-1]=a[n-1][n]/a[n-1][n-1]
for i in range(n-2,-1,-1):
x[i]=a[i][n]
for j in range(n-1,i,-1):
        x[i]=x[i]-a[i][j]*x[j]
        x[i]=x[i]/a[i][i]
print'The solution is '
for i in range(0,n):
        print'x[',i,']=',x[i]
```

5 Lagrange's Interpolation Formula

```
#Lagrange's Interpolation Formula
print'LAGRANGE INTERPOLATION FORMULA'
print'------------------------------'
p=int(input('Enter the number of X values: '))
x=[]
y=[]
for i in range(1,p+1):
        a=float(input('Input X values '))
        x.append(a)
print'***************'
for i in range(1,p+1):
        b=float(input('Input Y values '))
        y.append(b)
print'***************'
q=float(input('Enter the value at which interpolated '))
z=0
for i in range(0,p):
        k=1.0
        for j in range(0,p):
                if j!=i:
                        k=k*((q-x[j])/(x[i]-x[j]))
        z=z+(k*y[i])
print'Interoplated value at',q,' = ', z
```

6 Newton Divided Difference Interpolation

```
#Newton Divided Difference Interpolation
import numpy
from numpy import *
print('NEWTONS DIVIDED DIFFERENCE INTERPOLATION')
print('----------------------------------------')
x=[]
p=int(input('Enter the number of X values '))
y=zeros((p,p))
for i in range(0,p):
        c=input('Input X values ')
        x.append(float(c))
print('****************************************')
print('Input Y values ')
for i in range(0,p):
        y[i][0]=input()
print('****************************************')
z=float(input('Enter the value at which interpolated '))
print('****************************************')
for j in range(1,p):
        for i in range(0,p-j):
                y[i][j]=float((y[i+1][j-1]-y[i][j-1])/(x[i+j]-x[i]))
sum=y[0][0]
for i in range(1,p):
        t=1
        for j in range(0,i):
                t=t*(z-x[j])
        sum=sum+(t*y[0][i])
print'The interpolated value at x= ',z,' is ',round(sum,3)
```

7 A python program to approximate a root of a polynomial using the newton-raphson method

```
#A python program to approximate a root of a polynomial using the newton-raphson method
import math
import numpy
#f(x) - the function of the polynomial
def f(x):
        function = (x*x) - 3*(x)+ 2
        return function
def derivative(x): #function to find the derivative of the polynomial
        h = 0.000001
        derivative = (f(x + h) - f(x)) / h
        return derivative
def newton_raphson(x):
        return (x - (f(x) / derivative(x)))
#p - the initial point i.e. a value closer to the root
#n - number of iterations
def iterate(p, n): #
        x = 0
        for i in range(n):
                if i == 0: #calculate first approximation
                        x = newton_raphson(p)
                else:
                        x = newton_raphson(iterate(x, n)) #iterate the first and subsequ
                n=n-1
        return x
print'***************************'
print 'THE NEWTON RAPHSON METHOD'
print'***************************'
p=float(input('Enter the initial value '))
n=int(input('Enter the number of  iterations '))
print'***************************'
print 'the root of the polynomial x^2 - 3x + 2 using', n,' iterations and taking initial
```

8 A python programm for Runge Kutta methods 2 and 4 for solving Differential equation $x^2 + y^2$

```
# A python programm for Runge Kutta methods 2 and 4 for solving Differential equation x^2

from numpy import *
from math import ceil
import math
def f(x,y):
        return(x**2+y**2)
print'****************************'
print 'RUNGE KUTTA METHODS OF SOLVING DIFFERENTIAL EQUATIONS'
print'****************************'
x=float(input('Enter the initial conditions for x '))
y=float(input('Enter the initial conditions for y '))
print'****************************'
z=float(input('Enter the final value of the independent variable '))
print'****************************'
h=float(input('Enter the Enter the step length '))
print'****************************'
c=math.ceil((z-x)/h)
n=int(c)
m=int(input('Enter the order of the method (2 or 4) '))
if(m==2):
        print'Runge Kutta II order is used '
        for i in range(0,n+1):
                print'y(',x,')=',y
                k1=h*f(x,y)
                k2=h*f(x+(3.0/2)*h,y+(3.0/2)*k1)
                y=y+(2*k1+k2)/3.0
                x=x+h
else:
        print'Runge Kutta IV order is used '
        for i in range(0,n+1):
                print'y(',x,')=',y
                k1=f(x,y)
                k2=f(x+(1.0/2)*h,y+(1.0/2)*k1*h)
                k3=f(x+(1.0/2)*h,y+(1.0/2)*k2*h)
                k4=f(x+h,y+(k3*h))
                y=y+((k1+(2*k2)+(2*k3)+k4)/6)*h
                x=x+h
```

9 Gauss Siedel Iteration Method

```
# Gauss Siedel Iteration Method
import numpy as np
from scipy.linalg import solve

def gauss(A, b, x, n):
        L = np.tril(A)
        U = A - L
        for i in range(n):
                x = np.dot(np.linalg.inv(L), b - np.dot(U, x))
                print str(i).zfill(3),
                print(x)
        return x

'''___MAIN___'''
#modify from here
print'********************'
print 'GUASS SIEDEL ITERATION METHOD'
print'********************'
m=input('The number of equations ')
A=np.zeros((m,m)) #matrix creation for storing the values
print('Enter the coefficients  ')
for i in range(0,m):
        for j in range(0,m):
                print'A[',i,'][',j,'] :'
                A[i][j]=input()
b=np.zeros((m))
print'********************'
print('The constants  ')
for l in range(0,m):
        print'b[',l,'] :'
        b[l]=input()
print'********************'
x=np.zeros((m))
print('Enter the initial values  ')
for p in range(0,m):
print'x[',p,'] :'
x[p]=input()
print'********************'
n=input('The number of iterations ')
print'********************'
#A = np.array([[2.0, 6.0, -1.0], [5.0, -1.0, 2.0], [-3.0, -4.0, 1.0]])
#b = [-14.0, 29.0]
#x = [0, 0]
#n = 6
```

```
#print gauss(A, b, x, n)
w=solve(A, b)
print 'The solution is ',w
```

10 A python programm for simpson's method of integration

```
# A python programm for simpson's method of integration
#import math
from numpy import *
x=[]
y=[]
print'*****************************'
print 'THE SIMPSONS METHOD OF INTEGRATION'
print'*****************************'
p=int(input('Enter the number of X values '))
#a=zeros((p))
for i in range(0,p):
        c=input('Input X values ')
        x.append(float(c))
print('****************************************')
for i in range(0,p):
        d=input('Input Y values ')
        y.append(float(d))
print('****************************************')
a=x[0]
b=x[p-1]
h=(x[1]-x[0])
sum=y[0]+y[p-1]
for i in range(1,p-1):
        if(i%2==0):
                sum=sum+2*y[i]
        else:
                sum=sum+4*y[i]
integral=sum*h/3.0
#sigma=0.0
#sigma=(b-a)*(a+4*y[2]+y[p-1])/6
print'The integral of the function from ', a ,' to ', b ,' is ',integral
```

11 A python programm for trapezoidal's rule of integration

```
# A python programm for trapezoidal's rule of integration
print'TRAPEZOIDAL RULE OF INTEGRATION'
from numpy import *
x=[]
y=[]
print'****************************'
print 'THE TRAPEZOIDAL RULE OF INTEGRATION'
print'****************************'
p=int(input('Enter the number of  X values '))
#a=zeros((p))
for i in range(1,p+1):
        c=input('Input X values ')
        x.append(float(c))
print('****************************************')
for i in range(1,p+1):
        d=input('Input Y values ')
        y.append(float(d))
print('****************************************')
a=x[0]
b=x[p-1]
h=x[1]-x[0]
s=y[0]+y[p-1]
for i in range(1,p-1):
        s=s+2*y[i]
integral=(h/2)*s
print'The integral of the function from ', a ,' to ', b ,' is ',integral
```

12 Triangular Factorization of Square Matrices

```
#Triangular Factorization of Square Matrices
import numpy
import math
from numpy import *
import numpy as np
from math import fabs
print'********************'
print 'Triangular Factorization of a Invertible Square Matrix'
print'********************'
n=int(input('The order of the non-singular square matrix  '))
print'********************'
a = [[0] * n for i in range(n+1)]
c = [[0] * n for i in range(n+1)]
#x = [[0] * m for i in range(m-1)]
x=(zeros(n))
for i in range(0,n):
        print'Enter the coeficients of ',i, 'th row of the square matix '
        for j in range(0,n):
                z=float(input())
                a[i][j]=z
print'********************'
print'The given matrix is '
for i in range(0,n):
        for j in range(0,n):
                print '\t', a[i][j],
                print '\n'
print'********************'
for l in range(0,n):
        if(a[l][l]==0):
                for k in range(l+1, n):
                if(a[k][l]!=0):
                break
                for j in range(0,n+1):
                        s=a[l][j]
                        a[l][j]=a[k][j]
                        a[k][j]=s
        t=a[l][l]
#c[l][l]=1.0
        for m in range(l,n):
                c[m][l]=abs(a[m][l]/a[l][l])
        for j in range(0,n):
                a[l][j]=a[l][j]
                for i in range(l+1,n):
                        s=a[i][l]
```

```
                        for k in range(0,n):
                                a[i][k]=a[i][k]-s*a[l][k]/t
print'********************'
print'The upper triangular matrix of the given matrix is '
for i in range(0,n):
        for j in range(0,n):
                print '\t',a[i][j],
        print '\n'
print'********************'
print'The lower triangular matrix of the given matrix is '
for i in range(0,n):
        for j in range(0,n):
                if i<j:
                        c[i][j]=0
                if i==j:
                        c[i][j]=1
                print '\t',c[i][j],
        print '\n'
print'********************'
```