Find a positive real root of (x**4-x-10)=0 using newton Rapfson method.

```
def f(x):
  return x**4-x-10
def f1(x):
  return 4*x**3-1
X0=float(input("Enter the initial approximation"))
for i in range(1,10):
  Xn=X0-f(X0)/f1(X0)
  X0=Xn
print("The approimate root using newton Rapfson method is %.4f"%Xn)
```

```
Enter the initial approximation2
The approimate root using newton Rapfson method is 1.8556
```

Find a positive real root of (x**3-x-2)=0 using newton Rapfson method.

```
import math
def f(x):
  return 3*x-math.cos(x)-1
def f1(x):
  return 3+math.sin(x)
X0=float (input ("Enter the initial approximation: "))
for i in range (1,10):
  Xn=X0-f(X0)/f1(X0)
  X0=Xn
  print ("The approimate root using newton Rapfson method is %.4f"%Xn)
```

```
Enter the initial approximation: 1
The approimate root using newton Rapfson method is 0.6200
The approimate root using newton Rapfson method is 0.6071
The approimate root using newton Rapfson method is 0.6071
The approimate root using newton Rapfson method is 0.6071
The approimate root using newton Rapfson method is 0.6071
The approimate root using newton Rapfson method is 0.6071
The approimate root using newton Rapfson method is 0.6071
The approimate root using newton Rapfson method is 0.6071
The approimate root using newton Rapfson method is 0.6071
```

Solve the system of equation using Gauss-seidal method.

```
X0=0; Y0=0; Z0=0
for i in range (1,10):
  X=1/5*(12-2*Y0-Z0)
  X0=X
  Y=1/4*(15-X0-2*Z0)
  Y0=Y
```

```
  Z=1/5*(20-X0-2*Y0)
  Z0=7
print ("The approximate solution of X = %.4f, Y = %.4f,Z = %.4f"%(X,Y,Z))
```

```
The approximate solution of x=1.0000,Y=0.0000,Z=3.8000
```

```
Unit 4 : Langrange's Interpolation
```

```
x= [0,1,3,4]
y= [-12,0,6,12]
s = float(input("Enter the value of x to be in: "))
sum=0
for i in range (0,4):
  prod=1
  for j in range (0,4):
    if i!=j:
      prod = prod*(s-x[j])/(x[i]-x[j])
      sum = sum+prod*y[i]
print ("The functional value is %.4f"%sum)
```

```
Enter the value of x to be in: 0
The functional value is -36.0000
```

```
Trapizodal
```

```
def f(x):
  return 1/(1+x**2)
a=float(input("Enter the lower limit: " ))
b=float(input("Enter the upperlimit: "))
h=float(input("Enter the step size: "))
n=int((b-a)/h)
sum = 0
for i in range (1, n):
  sum = sum+f(a+i*h)
  trap = h/2*(f(a)+f(b)+2*sum)
print ("The Integral value is %.5f"%trap)
```

```
Enter the lower limit: 0
Enter the upperlimit: 1
Enter the step size: 0.1
The Integral value is 0.78498
```

```
R K method
```

```
def f (x, y):
  return x+y**2
x0=float(input("Enter initial point of x: "))
```

```
y0=float(input("Enter initial point of y: "))
h=float(input("Enter step value h: "))
k1 = h*f(x0, y0)
k2 = h*f(x0+h/2, y0+k1/2)
k3 = h*f(x0+h/2, y0+k2/2)
k4 = h*f(x0+h, y0+k3)
y = y0+(k1+2*k2+2*k3+k4)/6
print ("The value of y using RK method is %.4f"%y)
```

```
Enter initial point of x: 0
Enter initial point of y: 1
Enter step value h: 0.1
The value of y using RK method is 1.1165
```

Adam Bashforth method

```
import math
def f (x, y):
  return (x**2)*(1+y)
X0=float (input ("Enter x0: "))
yO=float (input ("Enter y0: "))
X1=float (input ("Enter X1: "))
y1=float (input ("Enter y1: "))
X2=float (input ("Enter X2: "))
y2=float (input ("Enter y2: "))
X3=float (input ("Enter x3: "))
y3=float (input ("Enter y3: "))
h=0.1
y4p = y3+(h/24)*(55*f(X3, y3)-59*f(X2, y2) +37*f(X1,y1)-9*f(X0, y0))
X4=x3+h
y4c = y3+(h/24)*(9*f(X4, y4p) +19*f(X3, y3) -5*f(X2, y2) +f(X1, y1))
print("Approximate soln is %0.4f"%y4c)
```

```
Enter x0: 1
Enter y0: 1
Enter X1: 1.1
Enter y1: 1.233
Enter X2: 1.2
Enter y2: 1.548
Enter x3: 1.3
Enter y3: 1.979
Approximate soln is 2.5749
```

+ Code + Text

Colab paid products - Cancel contracts here

1m 51s completed at 10:45 PM