422MA5077 Amisha Patel Assignment-II

```
#Write a python script to solve x**2 + y**2 - 4 = 0, x - y + 1 = 0.
!pip install sympy
from sympy import symbols, Eq, solve
x, y = symbols('x y')
eq1 = Eq(x**2 + y**2, 4)
eq2 = Eq(x - y, -1)
solutions = solve((eq1, eq2), (x, y))
print(solutions)
```

```
Requirement already satisfied: sympy in /usr/local/lib/python3.11/dist-packages (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy) (1.3.0)
[(-1/2 + sqrt(7)/2, 1/2 + sqrt(7)/2), (-sqrt(7)/2 - 1/2, 1/2 - sqrt(7)/2)]
```

```
#Using Newton-Raphson method, with 3 as a starting point, to find the value of sqrt.10 within an accuracy of 10^-8.
def f(x):
  return x**2 - 10
def df(x):
  return 2*x
def Newton_Raphson_method(f,df,xo,tol=1e-8,max_iter=100):
  for i in range(max_iter):
    x1 = xo-f(xo)/df(xo)
    if df(xo) == 0 :
        print("No solution possible for df=0")
        return None

    if abs(x1-xo)<tol:
      return x1
    xo=x1
  print("Maximum iteration reached without convergence")
  return None

xo = 3
root = Newton_Raphson_method(f,df,xo)
print(root)
print(f(root))
```

```
3.1622776601683795
1.7763568394002505e-15
```

```
#Explain what happens whe Newton's Method is applied to find the root of the equation:x**3 - 3*x + 6 starting with xo = 1.
def f(x):
  return x**3 - 3*x + 6
def df(x):
  return 3*x**2 -3
def Newton_Raphson_method(f,df,xo,tol=1e-8,max_iter=100):
  for i in range(max_iter):
    x1 = xo-f(xo)/df(xo)
    if df(xo) == 0 :
        print("No solution possible for df=0")
        return None

    if abs(x1-xo)<tol:
      return x1
    xo=x1
  print("Maximum iteration reached without convergence")
  return None

#xo = 1 for initial guess as 1 it is giving derivative = 0 for which solution is not possible.
xo = -2
root = Newton_Raphson_method(f,df,xo)
print(root)
print(f(root))
```

```
-2.35530139760812
-1.7763568394002505e-15
```

```
#Solve: x = tanx by bisection method.
import math
def f(x):
  return math.tan(x) - x
def bisection_method(f,a,b,tol=1e-8,max_iter=100):
```

```
  if f(a)*f(b)>=0:
    print("Change the initial guesses")
    return None
  for i in range(max_iter):
    c = (a+b)/2
    if f(c)==0 or (b-a)<tol:
      return c

    if f(a)*f(b)<0:
      b=c
    else:
      a = c
    print("Maximum iteration is reached.")
    return a+b/2

print(bisection_method(f,-0.1,0.1))
print(f(-0.1))
```

```
0.0
-0.0003346720854505436
```

```
#Using bisection method find a root of the following equation: xsinx - 1 = 0 in [0,2].
import math
def f(x):
  return x*math.sin(x) - 1
def bisection_method(f,a,b,tol=1e-8,max_iter=100):
  if f(a)*f(b)>=0:
    print("Change the initial guesses")
    return None
  for i in range(max_iter):
    c = (a+b)/2
    if f(c) == 0 or (b-a)/2 < tol:
      return c

    if f(c)*f(a)<0:
      b=c
    else:
      a = c
  print("Maximum iteration is reached.")
  return (a+b)/2

root=bisection_method(f,0,2)
print(root)
print(f(root))
```

```
1.114157147705555
9.490602304040863e-09
```

```
#Using bisection method find a root of the following equation: xsinx - 1 = 0 in [0,2].
import math
def f(x):
  return math.cos(x)- math.sqrt(x)

def bisection_method(f,a,b,tol=1e-6,max_iter=100):
  if f(a)*f(b)>=0:
    print("Change the initial guesses")
    return None
  for i in range(max_iter):
    c = (a+b)/2
    if f(c) == 0 or abs(b-a/2) < tol:
      return c

    if f(c)*f(a)<0:
      b=c
    else:
      a = c
  print("Maximum iteration is reached.")
  return (a+b)/2

root=bisection_method(f,0,1)
```

```
print(root)
print(f(root))
```

```
0.6417143708728826
0.0
```

```
#Find the positive root of tan(piex) - 6 = 0 in [0,0.48] using secant method.
import math
def f(x):
  return math.tan(math.pi*(x)) - 6
def secant_method(f,xo,x1,tol=1e-8,max_iter=100):
  if f(xo)*f(x1)>=0:
    print("chnage the initial guesses")
    return None
  for i in range(max_iter):
    f_xo = f(xo)
    f_x1 = f(x1)

    if f_x1 - f_xo == 0:
        print("Secant method failed: denominator is zero")
        return None
    x2 = x1 - f_x1*(x1-xo)/(f_x1-f_xo)
    if abs(x2-x1)<tol:
      return x2
    xo = x1
    x1 = x2
  print("Maximum iteration is reached.")
  return None

xo = 0.44
x1 = 0.48
root = secant_method(f,xo,x1)

if root is not None:
    print(root)
    print(f(root))
else:
    print("Root not found.")
```

```
0.44743154328874657
-5.329070518200751e-15
```

```
#A chemical firm produces q kg of a chemical per day for which it costs P(q) dollars. The cost-product relation can be written as the follow
   #P(q) = 100 + 2q + 3q^(2/3)
   #The firm sells any amount of chemical at rate of 5  dollars/kg. Find the break-even point of the firm,that is, how much it produce per d
def f(q):
    # The break-even equation f(q) = 100 - 3q + 3q**(2/3),when cost function = Revenue function =100+2q+3q*(2/3)=5q.

    return 100 - 3*q + 3*q**(2/3)
def df(q):
    return -3 + 2*q**(-1/3)
def newton_Raphson_method(f,df,q,tol=1.0e-6,max_iter=100):
    for i in range(max_iter):
        if df(q)==0:
            print("Derivative is zero; no solution exists.")
            return None
        q_new = q - f(q) / df(q)
        if abs(q_new - q) < tol:
            return q_new
        q = q_new
    print("Maximum iterations reached without convergence.")
    return None

initial_guess = 50
root = newton_Raphson_method(f, df, initial_guess)
if root is not None:
    print(f"The root is approximately: {root}")
else:
    print("The method failed to find a root.")
print(f(root))
```

```
The root is approximately: 46.2107474505418
2.1316282072803006e-14
```

```
#Consider the van der Walls equation of state:
#(P + a/v**2)(v - b) = RT
#Use Newton-Raphson method to compute the specific volume V of one mole gas at T = 300 K ,given P = 1,R = 0.082054J J(kg-K),a = 3.592 Pa-m**6,
#Estimate the initial approximation Vo from the ideal gas law PV = RT.
def f(V, P, R, T, a, b):
    # van der Waals equation rearranged: f(V) = (P + a/V**2)(V - b) - RT
    return (P + a / V**2) * (V - b) - R * T

def df(V, P, R, T, a, b):
    term1 = -(2 * a * (V - b)) / V**3
    term2 = (P + a / V**2)
    return term1 + term2

def newton_raphson(f, df, V0, tol=1e-6, max_iter=100, P=1, R=0.082052, T=300, a=3.592, b=0.04267):
    V = V0
    for i in range(max_iter):
        fV = f(V, P, R, T, a, b)
        dfV = df(V, P, R, T, a, b)
        if abs(fV) < tol:
            return V
        if dfV == 0:
            print("Derivative is zero. No solution found.")
            return None
        V = V - fV / dfV
    print("Maximum iterations reached. No solution found.")
    return None

P = 1
R = 0.082052
T = 300
a = 3.592
b = 0.04267

P = P * 101325
R = R * 101.325
# Initial guess using ideal gas law: V0 = RT/P
V0 = R * T / P
V = newton_raphson(f, df, V0, P=P, R=R, T=T, a=a, b=b)

if V is not None:
    print(f"The specific volume V is approximately: {V:.6f} m^3/mol")
else:
    print("Could not find a solution.")
```

```
The specific volume V is approximately: 0.067093 m^3/mol
```

Start coding or generate with AI.