Computational Methods and Modelling

Antonio Attili & Edward McCarthy

antonio.attili@ed.ac.uk
ed.mccarthy@ed.ac.uk

School of Engineering
University of Edinburgh
United Kingdom

Solution tutorial 3
Root finding.

THE UNIVERSITY of EDINBURGH

Exercise 2: Newton Raphson

- This is a possible way to implement Newton Raphson, including an error control strategy.

newton.py

```python
def newton(f,Df,x0,epsilon,max_iter):
    xn = x0
    for n in range(0,max_iter):
        fxn = f(xn)
        if abs(fxn) < epsilon:
            print('Found solution after',n,'iterations.')
            return xn
        Dfxn = Df(xn)
        if Dfxn == 0:
            print('Zero derivative. No solution found.')
            return None
        xn = xn - fxn/Dfxn
    print('Exceeded maximum iterations. No solution found.')
    return None

f = lambda x: x**2 + 4*x - 12
df= lambda x: 2*x + 4
x0=1
epsilon=0.0001
max_iter=100
solution = newton(f,df,x0,epsilon,max_iter)
```

Exercise 2: Secant

- This is a possible way to implement the secant method.

secant.py

```python
def secant(f,a,b,N):
    if f(a)*f(b) >= 0:
        print("Secant method fails.")
        return None
    a_n = a
    b_n = b
    for n in range(1,N+1):
        m_n = a_n - f(a_n)*(b_n - a_n)/(f(b_n) - f(a_n))
        f_m_n = f(m_n)
        if f(a_n)*f_m_n < 0:
            a_n = a_n
            b_n = m_n
        elif f(b_n)*f_m_n < 0:
            a_n = m_n
            b_n = b_n
        elif f_m_n == 0:
            print("Found exact solution.")
            return m_n
        else:
            print("Secant method fails.")
            return None
    return a_n - f(a_n)*(b_n - a_n)/(f(b_n) - f(a_n))

f = lambda x: x**2 + 4*x - 12
solution = secant(f,1,5,25)
print(solution)
```

Exercise 2: root finding using SciPy

- Root finding using the python library SciPy

root_scipy.py

```python
from scipy import optimize

# Definition of equation f(x)=0
# We want to find the roots of f(x)
def f(x):
    return x**2 + 4*x - 12

# First root
solution = optimize.newton(f, -10)
print(solution)

# Second root
solution = optimize.newton(f, 1.5)
print(solution)
```