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

Tutorial 3
Root finding.

THE UNIVERSITY
of EDINBURGH

Exercise 1: Root finding with the bisection method

```
def bisection(f,a,b,N):
    # Check if a and b bound a root
    if f(a)*f(b) >= 0:
       print("a and b do not bound a root")
       return None
    a_n = a
    b_n = b
    for n in range(1,N+1):
        m_n = (a_n + b_n)/2
        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("Bisection method fails.")
           return None
    return (a_n + b_n)/2

# we solve equation f(x)=0
f = lambda x: x**2 + 4*x - 12
# first root
approx_phi = bisection(f,-10,-3,5)
print(approx_phi)
# second root
approx_phi = bisection(f,0,10,5)
print(approx_phi)
```

Note: Code above is available on Learn.

Test the bisection method code shown on the left:

- Compare the results with the exact solutions obtained with the classical method for quadratic equations.
- Try to change the number of iteration N and assess the effect on the error with respect to the exact solution.
- Apply the bisection code to the equation $f(x) = sin(x) * e^{x^{0.1}}$ to find the smallest positive (non-zero) root.

Exercise 2: Newton Raphson, secant, and pyhton modules

Consider again the quadratic equation $f(x) = x^2 + 4x - 12 = 0$:

- Include an error control methodology in the bisection method code: The iterations in the code should be interrupted when a prescribed error is achieved. Consider an appropriate estimate of the error that does not require the exact solution.
- Implement a python code to find the roots of the equation with the Newton Raphson and with the secant methods, including an error control strategy.
- Implement a python code to find the roots using intrinsic python functions included in the module SciPy. See for example:
  `docs.scipy.org/doc/scipy/reference/optimize.html`
  and more specifically
  `docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.newton.html`