9 Root finding

Solve all problems using numerical and scientific python packages and show the solutions in jupyter Notebook. References:

Chapra & Canale. (2010). Numerical Methods for Engineers, 6th edition. Part two: Roots of equations.

Kiusalaas. (2013). Numerical Methods in Engineering with Python 3. Third Edition. Ch 4. Roots of Equations.

Johansson. (2015). Numerical Python: A Practical Techniques Approach for Industry. Ch. 5. Equation Solving.

1. Draw a graphical presentation of the following functions within a domain $x \in [-2, 2]$ and observe how many roots ($f(x) = 0$) each function has and where they approximately locate

   (a) $f_1(x) = x^2 - x - 1$

   (b) $f_2(x) = x^3 - 3\sin(x)$

   (c) $f_3(x) = e^x - 2$

   (d) $f_4(x) = 1 - x^2 + \sin(\frac{50}{1+x^2})$

2. Write a function that incrementally search the root of a function. It should use 4 arguments, f the function, a the lower and b the upper limits of the search domain, and dx the step of the increment. The function returns the estimate for the root, x_0, and the number of steps, n, needed to find the root

   ```
   def incrementSearch(f, a, b, dx):
       # Your code here
       return x0, n
   ```

   Test your incremental search function with the example functions given in problem 1. How many incremental steps are needed to find the first root with the step size of 0.01?

3. Study the `bisect()` function found in module `scipy.optimize`. Use that function to find the roots of the problem 1. Use tolerance value of 0.01. How do these results differ from the results obtained in previous problem?

4. Write your own function that implements the bisectional search method. The definition of the function should be the following:

```
def bisectSearch(f, a, b, tol, maxiter):
        # Your code here
        return x0, n
```

where f is the function, a is the lower and b is the upper limit of the search domain, *tol* is the tolerance in the result, and *maxiter* is the maximum number of iterations allowed. The function returns the estimate for the root, x_0, and the number of iterations, n.

Test your code with the mathematical functions given in problem 1 and compare the results with those obtained in problem 3. Are they similar?

5. Study the function `newton()` found in module `scipy.optimize`. Use that function to find the roots of the problem 1. Use tolerance value of 0.01. Try different initial estimate values x_0. Compare the results to the previous results. What do you observe?

6. Write your own code that implements the Newton-Raphson method. The definition of the function should be

```
def newtonSearch(f, x0, tol, maxiter):
        # Your code here
        return x0, n
```

where f is the function, x_0 is the initial estimate of the root, *tol* is the tolerance allowed in the result, *maxiter* is the maximum number of iterations allowed in search. The function should return the estimate for the root, x_0, and the number of iterations. The code should estimate the derivative of the function numerically $f'(x) = (f(x+h) - f(x))/h$, where h is a small number.

Test that function with the examples given in problem 1 and compare the results to previous results.

7. (BONUS) By combining the incremental, bisectional and Newton-Raphson methods write a code that finds multiple roots of a function within a given domain. The code should use inputs: f the function, a and b the limits of the domain, *tol* the tolerance of the root estimates, dx the step size for incremental search, and *maxiter* the maximum iterations in bisectional or Newton-Raphson iterations.

   Test your code with the examples given in problem 1 and compare the results to the previous results.