Solving nonlinear equations

Due: Tuesday, February 7, 11:59pm.

Push your assignment solutions to the appropriate GitHub Classroom repository. Submit the following for this assignment:

- A PDF document, typeset in LaTeX, that contains your solutions to questions 1(a), 1(b), 1(c), 1(g), 2(d) and 3(b).
- Files called `NewtonRaphsonA2.py` and `bisectionA2.py`, that contain the Python functions for performing Newton-Raphson iteration and the bisection method, respectively, as well as a file called `Quest1.py` that calls the functions defined in `NewtonRaphsonA2.py` and `bisectionA2.py` and plots the convergence properties of the two methods
- the plot, with labels, produced by `Quest1.py`
- A file called `EulerChebA2.py` that contains a Python function that performs Euler-Chebyshev iteration, as well as a file called `Quest2.py` that calls the Python functions defined in `NewtonRaphsonA2.py` and `EulerChebA2.py` and plots the convergence properties of the two methods
- the plot, with labels, produced by `Quest2.py`
- A file called `Quest3.py` that calls the function defined in `NewtonRaphsonA2.py` and plots the convergence properties of the problem described in Question 3.
- the plot, with labels, produced by `Quest3.py`

Make sure all files submitted include a comment line with your name and student number. Also, include comments with a brief description of the functionality, input and output arguments and usage of each function or script. Also, add some comments that explain what steps are taken. A SLACK channel for discussion on this assignment is available.

Question 1 — 25 marks

Consider the nonlinear equation

$$f(x) = e^{-x} + \cos(x+1) - 1 = 0$$

(a) Use the Intermediate Value Theorem to show that the equation $f(x) = 0$ has at least one solution on the interval $[0, 1]$.

(b) Use Rolle's Theorem (which is a special case of the Mean Value Theorem) to show that the equation $f(x) = 0$ has at most one solution on the interval $[0, 1]$. Hint: Use contradiction. That is, assume that there are two solutions, then show that $f(x)$ cannot satisfy the conditions of the theorem.

Note: Part (a) and (b) together show that $f(x) = 0$ has *exactly* one solution in the interval $[0, 1]$.

(c) Write out the Newton-Raphson iteration formula for finding the approximate solution of $f(x) = 0$, i.e. find the formula that gives $x^{(k+1)}$ as a function of $x^{(k)}$.

(**d**) Write a Python function that computes the approximate solution of an equation $f(x) = 0$ in a given interval using bisection, and returns the approximation and an array that contains the current approximation, the approximate errors and residuals for all iterations. Use the `bisectionA1.py` code as a starter code. See further instructions on the output variables in the starter code. Hint: we've already done this in class.

(**e**) Write a Python function that computes the approximate solution of an equation $f(x) = 0$ given an initial guess using the Newton-Raphson method, and returns the approximate solution and an array that contains the current approximation, the approximate errors and residuals for all iterations. Use the `NewtonRaphsonA1.py` code as a starter code. See further instructions on the output variables in the starter code.

(**f**) Write a script (Python code) called `Quest1.py` that

- uses your bisection code to find the approximate solution of the equation given above to a tolerance of less than 10^{-12}, with the appropriate starting interval.
- uses your Newton-Raphson method code to find the approximate solution of the equation given above to a tolerance of less than 10^{-12}, with a starting guess of your choice.
- plots the approximate errors for the bisection and Newton-Raphson methods versus the number of iterations on a semilogarithmic scale (log on the y axis). Hint: the bisection data should appear as a straight line. Make sure you label your plot appropriately.

(**g**) Which method is better if you want to minimize the number of computations to find an approximate solution with this accuracy? Explain.

Question 2 — 25 marks

We saw that the Newton-Raphson method converges very fast for approximating solutions of $f(x) = 0$. It can be derived from a linear (tangent line) approximation of $f(x)$ about the guess $x^{(k)}$, which involves computing $f'(x)$. You might ask whether we could derive a method with even faster convergence if we also incorporated the second derivative $f''(x)$. This can be done with the following iteration formula, which is sometimes referred to as the *Euler-Chebyshev method*:

$$x^{(k+1)} = x^{(k)} - \frac{f}{f'} - \frac{f''}{2f'}\left(\frac{f}{f'}\right)^2$$

where $f = f(x^{(k)})$, $f' = f'(x^{(k)})$, and $f'' = f''(x^{(k)})$, i.e. the function and its first and second derivatives, respectively, evaluated at $x^{(k)}$. It can be shown that this method shows cubic convergence, i.e. errors decrease as the cube of the error on each iteration.

(**a**) Using your Newton-Raphson Method code as a starting point, write a Python function that uses the Euler-Chebyshev method to find approximate solutions of $f(x) = 0$, given an initial guess. Hint: you will need to include an additional input that passes a Python function that computes $f''(x)$. Use `EulerCheb.py` as a starter code.

(**b**) Write a script (Python code) called `Quest2.py` that uses your code to approximate the solution of

$$f(x) = e^{-x} + \cos(x+1) - 1 = 0$$

to an error of less than 10^{-14}, with a starting guess of $x_0 = 1$.

(**c**) Also, in the script `Quest2.py`, compare your results to those obtained using your Newton-Raphson method code of Question 1, this time approximating the solution to an error of less than 10^{-14}, with the same starting guess as you used for the Euler-Chebyshev method. In particular, plot the approximate errors for the Euler-Chebyshev and Newton-Raphson methods versus the number of iterations on a semilogarithmic plot (the y axis should be in logarithmic scale). Make sure you label your plot appropriately.

(**d**) In this case, is the benefit of the Euler-Chebyshev method worth the extra work? Explain.

Question 3 **25 marks**

Consider the nonlinear equation

$$4x^4 - 8x^3 + 7x^2 - 3x + \frac{1}{2} = 0$$

(**a**) Write a script (Python code) called `Quest3.py` that

- uses your Newton-Raphson method code to find the approximate solution to the given equation to an error of less than 10^{-8}, with a starting guess of $x^{(0)} = 1$.
- plot the approximate errors for Newton-Raphson's method versus the number of iterations on a semilogarithmic scale (log on the y axis). Make sure you label your plot appropriately.

(**b**) Do the Newton-Raphson iterations converge quadratically (i.e. compare to the results found in Question 1)? What feature of the function do you think leads to the unexpected results. Hint: plot the function $f(x)$ vs. x near the approximate solution.