Bisection, Newton and secant

What to submit for this assignment?

- A PDF document, typeset with LaTeX, that contains the answers to questions 1(a), 1(b), 1(c), 1(g) and 2(c).
- A function called `bisect.py`, based on a bisection code in the `course_codes` repository, that returns a list of errors. Use the template in the assignment repository.
- A function called `Newton.py`, based on the Newton code in the `course_codes` repository, that returns a list of errors. Use the template in the assignment repository.
- A script called `Q1.py` that calls the two functions above and produces a plot showing the sequence of errors for bisection and Newton iteration.
- A function called `secant.py`. Use the template in the assignment repository.
- A script that calls the secant function to answer question 2(b) and produces the plot of question 2(c).

Question 1 **50 marks**

Consider the function

$$f(x) = e^{-x} - x = 0$$

(**a**) Can you "solve the equation by hand", i.e. find an explicit expression for x^* so that $f(x^*) = 0$?

(**b**) Use the Intermediate Value Theorem to show that the equation $f(x) = 0$ has at least one solution on the domain $[0, 2]$.

(**c**) Use the Mean Value Theorem to show that the equation $f(x) = 0$ has at most one solution on the domain $[0, 2]$.

From parts **b** and **c** we know that there is a unique solution x^* in $[0, 2]$.

(**d**) Change one of the bisection codes in the `course_codes` repository to return a list of errors.

(**e**) Change the Newton iteration code in the `course_codes` repository to return a list of errors.

(**f**) Write a script that

- uses your bisection code to find an approximation to x^* with an error less than 10^{-12};
- uses your Newton iteration code to find an approximation to x^* with an approximate error less than 10^{-12};
- plots the (approximate) error of bisection and Newton iteration versus the number of iterations on a semilogarithmic scale. The bisection data must appear as a straight line.

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

Question 2 **50 marks**

Consider the snowball problem from lecture 2. The function on slide 7/19 (point 2.) is implemented in the Python function `hor_dist.py`. Assume the friction coefficient is $c = 0.001$, the initial speed is $v_0 = 10.0\,(m/s)$ and the acceleration of gravity is $g = 9.81\,(m/s^2)$.

(**a**) Implement the secant method. It should return the approximate solution and a list of approximate errors.

(**b**) Assume your friends are standing at a distance of $R = 9(m)$. Wite a script that

- uses the secant function to find at what angle should you throw your snow ball to hit your friends (make sure your approximate solution is in the range $[\pi/4, \pi/2]$ and the approximate error of your answer is less than 10^{-12});
- plots the errors on a semilogarithmic scale.

(**c**) Compare the plot to that of question 1, part (f). Does the secant method converge like bisection or like Newton iteration?

Bonus **10 marks**

Save the figures of question 1(f) and 2(b) and include them in your LaTeX document. Add the following elements:

- Axis labels: "number of iterations" and "(approximate) error" and a legend to distinguish the bisection and Newton data. You can use simple commands from the `matplotlib` library for this purpose.
- A caption under the figure in your LaTeX document. Make sure the figure and the caption are centered and the caption briefly explains what data are shown and what you can conclude from them. Hint: use

```
\begin{figure}\begin{center} <command to include picture>
   \caption{...}\end{center}\end{figure}
```

.