FEEDBACK ON MAS2008 ASSIGNMENT 1 (ROOT FINDING)

Marks for this assignment were high, with an average close to 70%. A lot of that is due to a genuinely good effort by many students, for which you should be congratulated. However, to some extent it was due to a mark scheme that was not as well designed as it could have been. It is likely that the marks will be calibrated downwards as part of the usual adjustment process at the end of the semester, but I hope that that effect will not be too strong. I will try to make a better mark scheme for the next two assignments.

Marks (out of a total of 100) were assigned as follows:

- There were 5 marks each for defining the functions `newton()` and `secant()`.
- There were 20 marks in total for passing a series of automated tests. These used the eight test cases specified in the briefing together with four more cases. In more detail:
  - 3 marks for the Newton-Raphson method in cases 1,2,5,8,9 and 12 where the method succeeds with no issues.
  - 3 marks for the Newton-Raphson method in cases 3,6 and 11 where the method leaves the specified interval.
  - 3 marks for the Newton-Raphson method in cases 4 and 10 where the method fails due to division by zero.
  - 3 marks for the Newton-Raphson method in case 7 where the method loops and does not converge.
  - 3 marks for the secant method in cases 1,2,4,6,9 and 12 where the method succeeds with no issues.
  - 3 marks for the secant method in cases 3,5,7,8 and 11 where the method leaves the specified interval.
  - 2 marks for the Newton-Raphson method in case 8 where the method fails due to division by zero.
- There were 20 marks in total for general quality of diagrams. Credit was given for points including the following:
  - Clear labels, properly typeset in LaTeX, specifying the method used and the function.
  - Tangent lines and secant lines.
  - Points $(x_n, 0)$ and $(x_n, f(x_n))$ with vertical lines between them, and attached labels.
  - Intelligent choice of which lines and labels to include, so as to give an uncluttered picture which makes the operation of the method clear.
  - Zoomed-in pictures to show what happens as the method approaches the root.
  - Additional annotation for cases where either method has some special behaviour.
- There were 50 marks in total for commentary on the various test cases. Credit was given for points as listed below (but it was not necessary to mention all of the listed points to get full marks).
  - $x^2 - 1$ on $[0, 3]$ (4 marks): The exact roots are ± 1 with only 1 in the specified interval. Both methods succeed, but Newton's method requires fewer iterations. The iterations in Newton's method are given by $x_n = (1 + 5^{-2^n})/(1 - 5^{-2^n})$, as one can check by induction.
  - $e^{-x} - x$ on $[0, 3]$ (4 marks): The function is strictly decreasing, positive at $x = 0$ and negative at $x = 3$, so there is a unique root in $[0, 3]$. The root is approximately 0.5671432904, and can be described as $W(1)$, where W is the Lambert W-function. Both methods successfully find this root, but Newton's method requires fewer iterations.
  - $x^2 - 2x + 1.1$ on $[0, 3]$ (7 marks): There are no real roots, as one can easily see using the quadratic formula or by rewriting the function as $(x - 1)^2 + 0.1$. When x_n is not too close to the local minimum at $x = 1$, Newton's method will move towards $x = 1$, sometimes overshooting. When x_n is very close to $x = 1$, the denominator $f'(x_n)$ in Newton's method will be small, so x_{n+1} will be far from x_n and may lie outside the specified interval. On the other hand, the secant method

leaves the specified interval at the first step, as is always the case with the secant method when $f(a)$ and $f(b)$ have the same sign.

- $x^3 - 6x^2 + 12x - 6$ on $[0, 6]$ (7 marks). Newton's method fails with a division by zero error because $f'(x_1) = f'(2) = 0$ (for full credit, an explanation of why $x_1 = 2$ was also required). However, there is a unique root at $x = 2 - 2^{1/3} \approx 0.740078950$, and the secant method finds this.
- $\sin(x)$ on $[2, 18]$ (7 marks): The roots in this interval are $\{\pi, 2\pi, 3\pi, 4\pi, 5\pi\}$. Newton's method starts at $x_0 = 10$ and converges quickly to the nearby root at $3\pi \approx 9.424777962$. The secant method leaves the specified interval with $x_3 \approx 42.47716856$. For full credit, an explanation of the steps leading to x_3 and/or some geometric explanation of the flatness of the secant line is required. To analyse the eventual behaviour of the secant method accurately, one needs to calculate to at least 10000 digits of precision (which one could do using the Python `decimal` package). The sequence wanders around randomly for a long time, but after about 450 steps it starts converging rapidly to -24660632751539897π.
- $\cos(x)$ on $[2, 18]$, with the derivative specified incorrectly as $\cos(x)$ (7 marks): The roots in this interval are $\{\frac{3}{2}\pi, \frac{5}{2}\pi, \frac{7}{2}\pi, \frac{9}{2}\pi, \frac{11}{2}\pi\}$. The secant method successfully finds the root at $\frac{7}{2}\pi$. There is some luck involved in this: with slightly different parameters, the secant method could have struggled as in the previous case. However, we find that $3\pi < x_4 < x_3 < 4\pi$ with $\cos(x)$ increasing from -1 to $+1$ on $[3\pi, 4\pi]$; from this point on the method will converge to $\frac{7}{2}\pi$ with no trouble. On the other hand, with $f'(x)$ specified incorrectly as $\cos(x)$, the Newton-Raphson iteration is just $x_{n+1} = x_n - 1$ with $x_0 = 10$, so $x_n = 10 - n$, so x_9 lies outside the specified interval.
- $x^2 - 5x + 13$ on $[0, 8]$ (7 marks): There are no real roots, as one can easily see using the quadratic formula. The secant method leaves the specified interval at the first step, as is always the case with the secant method when $f(a)$ and $f(b)$ have the same sign. In Newton's method, we find that x_n is 4 when n is even and 1 when n is odd (more detailed equations and pictures are required for full credit). Thus, the method does not leave the interval but also does not converge.
- $|x - 3| - 2$ on $[0, 4]$ (7 marks): the unique root in this interval is at $x = 1$. Because the relevant part of the graph is a straight line, Newton's method gives the root exactly after the first step. In the secant method we have $x_0 = 0$ and $x_1 = 4$ and $x_2 = 2$. This gives $f(x_1) = f(x_2) = -1$ but the formula for x_3 involves division by $f(x_2) - f(x_1)$ so the method fails at this point. With a slightly different choice of interval, the secant method would succeed, but not very quickly.

Some additional comments:

- The test cases considered here involve functions that are given by a formula that can be evaluated very easily. However, one often wants to find roots of a function $f(x)$ where one can only evaluate $f(x)$ by some elaborate procedure, such as solving a differential equation, finding the eigenvalues of a large matrix, or some other strenuous computation. For this reason one should avoid calling $f(x)$ with the same argument x more than once. Instead of

```
if np.abs(f(x0)) < epsilon:
    return x0
x1 = x0 - f(x0)/df(x0)
```

  you should have

```
y0 = f(x0)
if np.abs(y0) < epsilon:
    return x0
x1 = x0 - y0/df(x0)
```

- For the same reasons, actual execution time is not the best way to compare the two methods: it is better to count the number of iterations, or the number of evaluations fo $f()$ and $f'()$. If you do want to calculate execution times, it is more reliable to use `timeit()` rather than `time()` or `perf_counter()`.

- The accuracy of the root returned by the two methods is not a useful criterion for comparison. The accuracy is largely controlled by the parameter `epsilon`, and any extra accuracy beyond that is determined by random features of the problem.
- The correct way to report an error condition is with the keyword `raise` followed by an exception constructor such as `RuntimeError("left the interval")`. It is not appropriate to use the keyword `return` in place of `raise`, or to simply print a message.
- The briefing specified that the root-finding methods should just return the root (and also that it was important to follow the specification precisely). Marks were deducted for root-finding functions that did not just return the root.
- Consider a case where $f(a)$ and $f(b)$ have the same sign (for example, where $f(x) = \cos(x)$ with $[a, b] = [0, 3]$). In this case, there may or may not be any roots in the interval $[a, b]$ ore elsewhere. If roots exist, then Newton's method will probably find them (possibly after leaving the interval $[a, b]$). The secant method will always leave the interval immediately, but it may re-enter the interval later, and it may also find a root if one exists. Some students started by checking whether $f(a)$ and $f(b)$ have the same sign, and raising an error if that was the case. This is certainly inappropriate for Newton's method. For the secant method it is more arguable, but to comply with the specification you should instead raise an error at the next stage reporting that the iteration has left the interval.
- Some students also checked at the beginning that a and b are real numbers with $a < b$, and that the other parameters looked sensible in similar ways. There are arguments for and against this, organised around the slogans "Look before you leap" and "It is easier to ask for forgiveness than permission". You can read more at `https://realpython.com/python-lbyl-vs-eafp/`.