Remember to adequately label all plots and include any requested code listings with your solutions. *Only include those scripts and functions which are requested.* A clear and complete presentation of your solutions is required for full credit.

1. Modify the provided `newton` function to display an appropriate error message when division by zero occurs or the maximum number of iterations is exceeded. Include the code for your modified function.

2. *Accelerating Newton's Method Near Simple Roots.* One way to speed up the convergence of Newton's method is by using the Halley iteration formula

$$g(x) = x - \frac{f(x)}{f'(x)}\left(1 - \frac{f(x)f''(x)}{2\,(f'(x))^2}\right)^{-1}.$$

The term in brackets is the modification of the Newton-Raphson formula.

(a) Using the provided `newton` function as a starting point, write a function `halley` which implements this iteration. You will need an additional input `fpp` for a function handle to the second derivative $f''(x)$. Include the code for your function.

(b) Using $f(x) = x^2 - 5$ and $x_0 = 2$, creating a table of values which demonstrates that Halley's method exhibits cubic convergence at simple roots of $f(x)$. The first column contains the iteration number i. The second column contains the error ratio $\varepsilon_{i+1}/\varepsilon_i^2$. The third column contains the error ratio $\varepsilon_{i+1}/\varepsilon_i^3$. The fourth column contains the error ratio $\varepsilon_{i+1}/\varepsilon_i^4$. Note that the method should converge after just a few iterations. All numerical values should be printed to at least 10 digits. Explain briefly how your data demonstrate cubic convergence.

(c) Discuss the trade-offs associated with using this method instead of Newton's method. That is, why would you want to use one instead the other and vice versa?

3. *Accelerating Newton's Method Near Multiple Roots.* If the order of a root is known, it is possible to avoid the loss of quadratic convergence associated with a multiple root in the following way. Recall that if r is a root of order m, then the k-th derivative $f^{(k)}(r) = 0$ for all $k < m$.

(a) Suppose that r is a root of $f(x)$ with order m. Prove that the modified Newton-Raphson iteration

$$x_k = x_{k-1} - \frac{m\,f(x_{k-1})}{f'(x_{k-1})}$$

converges quadratically. Also find the limiting error ratio $\varepsilon_{i+1}/\varepsilon_i^2$ as $i \to \infty$.

(b) Discuss the trade-offs associated with using this method instead of Newton's method. That is, why would you want to use one instead the other and vice versa?

4. *Accelerating Newton's Method Near Multiple Roots, Part 2.* Here is another way of accelerating Newton's method that does not require knowing the order of a root ahead of time. Recall that if r is a root of order m, then $f(x) = (x-p)^m q(x)$ with $q(r) \neq 0$.
   (a) Show that $h(x) = f(x)/f'(x)$ has a simple root at r.
   (b) Show that when the Newton-Raphson iteration is applied to finding the simple root r of $h(x)$ we get $g(x) = h(x)/h'(x)$ which becomes
   $$g(x) = x - \frac{f(x)f'(x)}{(f'(x))^2 - f(x)f''(x)}.$$
   (c) The iteration using $g(x)$ converges quadratically to r. Explain why this happens.
   (d) Discuss the trade-offs associated with using this method instead of Newton's method. That is, why would you want to use one instead the other and vice versa?

5. *Order of Convergence for the Secant Method.* Recall that if an iterative method converges with order p, we have
   $$\frac{\varepsilon_{i+1}}{\varepsilon_i^p} \approx M \quad \text{and} \quad \frac{\varepsilon_{i+2}}{\varepsilon_{i+1}^p} \approx M$$
   as $i \to \infty$ for some rate $M < \infty$.
   (a) By setting the two ratios equal to each other and taking logarithms, show that
   $$p \approx \frac{\log \varepsilon_{i+2} - \log \varepsilon_{i+1}}{\log \varepsilon_{i+1} - \log \varepsilon_i},$$
   thus giving us a procedure to estimate p.
   (b) Using the provided `newton` function as a starting point, write a function `secant` which implements the secant method. You will need an additional input `x1` for the second initial guess, and you won't need the input `fp`. The objective function `f` should only be evaluated once per iteration. Include the code for your function.
   (c) Devise an experiment using the estimate above to show that the secant method converges with order $p = (1+\sqrt{5})/2 \approx 1.6180$ to a simple root of $f(x)$. Include your objective function, initial guesses, and a table of your estimates of p at each iteration.