Whenever calculations are needed to solve a problem, those calculations must be submitted as part of the homework assignment.

Homework must be submitted electronically. Unless express permission has been given in advance by the instructor for a late homework submission, a 30% percent penalty will be deducted for each late day (or part of a late day).

Exercise 2.1. This problem is designed to provide experience with Richard Brent’s \texttt{fzero}, probably the most used zero- finding algorithm in the world, which is available through Matlab and Octave. The algorithm in \texttt{fzero} generates each new iterate using a combination of secant, inverse quadratic interpolation, bisection, and “a step of $\delta$” (a very small step). A “textbook version” of \texttt{fzero}, called \texttt{fzerotx.m}, written in Matlab, can be found at \url{mathworks.com/moler/chapters.html\rightarrow NCM\rightarrow Textbook files\rightarrow fzerotx.m}

Using \texttt{fzero} in Matlab to find a zero of the function $f(x)$ involves a function that must be named “fun”, where \texttt{fun(x)} computes $f(x)$. Suppose that your Matlab function-evaluation code is done by the function \texttt{ftest}, with the following code in \texttt{ftest.m}:

```matlab
function y = ftest(x);
% function for use with Matlab fzero
global fcount;
y = [calculation of f(x)];
fcount = fcount + 1;
[print x and y]
end
```

In the above, \texttt{fcount} is declared as a global variable (and should be declared as such in the main program) so that it can be updated within a function subroutine.

To apply \texttt{fzero} to your function, execute the Matlab commands:

```matlab
fun = @ftest
xstar = fzero(fun,x0)
```

In calling \texttt{fzero}, two options are available for setting $x_0$: (1) $x_0$ can be a single point, e.g., $x_0 = 1$; or $x_0$ can be an interval of the form $x_0 = [a \ b]$, where $f$ has opposite signs at $a$ and $b$. In this problem, you will call \texttt{fzero} with $x_0$ defined as a single point.

When called with a single point, \texttt{fzero} generates a sequence $\{x_k\}$ in which it first tries to “find” two iterates where $f$ has opposite signs; these define a so-called “straddle”, also called an “interval of uncertainty”. Once such an interval has been found, the logic reverts to Brent’s algorithm.

In this problem, the function whose zero is to be found is $f(x) = x^3 - 8$, but the main point is for you to perform numerical reverse-engineering by deducing how \texttt{fzero} tries to obtain an interval in which $f$ changes sign, starting with a single point $x_0$. Try three different choices for $x_0$:

(a) $x_0 = -1$;  
(b) $x_0 = 4$;  
(c) $x_0 = 0$.

Your code should print every evaluation of $f$, giving $x$, the value of $f(x)$, and the running total of the number of times $f$ has been evaluated. You may like to experiment with other functions and other values of $x_0$. 
(i) Based on your experiments, what algorithmic strategy is being used by \texttt{fzero}, starting with a single point $x_0$, to locate an interval in which $f$ changes sign?

(ii) Do you think that the \texttt{fzero} strategy is effective, meaning that it finds an interval of uncertainty with a “small” number of function evaluations? Explain why or why not.

(iii) Can you find a function $f$ and a starting point $x_0$ for which \texttt{fzero} fails to locate an appropriate interval when one exists? If so, explain exactly how and why the \texttt{fzero} strategy fails.

(iv) Give your own algorithm for finding an interval in which $f$ changes sign, and try it on $f$ with starting points from (a)–(c), above. Comment on how well (or not) it performs, in terms of quickly finding an interval where $f$ changes sign. How does its performance (meaning the number of function evaluations needed to find the interval) compare with the performance of \texttt{fzero}?

(v) Explain any disadvantages of your method, including situations in which it might fail to find an appropriate interval.

\textbf{Exercise 2.2.} For this problem, you will use your already-written programs for Newton’s method and the secant method, which are designed to find $x^*$ such that $f(x^*) = 0$, where $f$ is a scalar-valued twice-continuously differentiable function of a single real variable.

In addition, write a program to implement Wheeler’s bent secant method, where the inputs include two starting guesses and a subroutine that calculates $f(x)$ for a given argument $x$. Wheeler’s method starts with an interval $[a, b]$ where we require that $f(a)f(b) < 0$. Each subsequent iteration computes a new iterate by fitting a straight line using the $[x, y]$ pairs $[a, fa]$ and $[b, fb]$, and the new iterate becomes $a$ or $b$ at the next iteration. However, because Wheeler’s method may use a “fake” function value related by means of the current “Wheeler factor” to the function value at a previous iterate, it will not necessarily be true that $fa$ is the value of $f$ at $a$, and a similar caution applies for $fb$.

All your codes should stop (i) if an exact zero $\bar{x}$ is found, i.e., such that $f(\bar{x}) = 0$, (ii) after \texttt{maxit} iterations, and (iii) if $|f|$ is less than \texttt{ftol}, for specified values of \texttt{maxit} and \texttt{ftol}.

The function of interest is

$$f(x) = x^3 - 8.$$  \hfill (2.1)

Figure 2.1 shows the graph of this function in an interval around the origin. The graph may be helpful in understanding some of the behavior that you will observe.
3

(a) Run the Newton code with two starting points: (i) \( x_0 = 1 \), with \( \text{maxit} = 12 \) and \( \text{ftol} = 10^{-14} \); and (ii) \( x_0 = -0.1 \), with \( \text{maxit} = 20 \) and \( \text{ftol} = 10^{-14} \).

At iteration \( k \) in the Newton code, print \( k, x_k, f_k, f'_k \), and \( x_k - x^* \), using scientific notation and showing at least 7 digits of accuracy. In addition, please count and print the number of times that \( f \) is evaluated before termination, including the evaluation of \( f(x_0) \).

For each starting point, comment on the behavior of the Newton iterates, explaining what happens to the Newton iterates and why. With starting point (ii), comment in particular on the behavior of the iterates during the first 12 iterations.

(b) Run the secant code, with \( \text{maxit} = 25 \) and \( \text{ftol} = 10^{-14} \), for these three pairs of initial points:

(i) \( x_0 = -0.6 \) and \( x_1 = 4 \); (ii) \( x_0 = 1 \) and \( x_1 = 5 \); (iii) \( x_0 = 5.6 \) and \( x_1 = 0.1 \). At iteration \( k \) in the secant code, print \( k, x_k, f_k, x_{k+1}, f_{k+1}, x_{k+1} - x^* \), and the ratio \( |x_{k+1} - x^*|/|x_k - x^*| \), using scientific notation and showing at least 7 digits of accuracy.

Comment on the behavior of the iterates in each case—for example, does convergence seem to be superlinear? What about the "waltz"? gv

(c) Run the Wheeler code, with \( \text{maxit} = 25 \) and \( \text{ftol} = 10^{-14} \), for the same three pairs of initial points as in part (b): (i) \( x_0 = -0.6 \) and \( x_1 = 4 \); (ii) \( x_0 = 1 \) and \( x_1 = 5 \); (iii) \( x_0 = 5.6 \) and \( x_1 = 0.1 \). At iteration \( k \) of Wheeler’s method, print \( k \) and the values \( a, f_a, b, f_b \) that define the \([x, y]\) pairs \([a, f_a]\) and \([b, f_b]\) used to compute the next iterate. Print the “Wheeler factor”, which is denoted by \( \mu \) in Algorithm 6.1 (Lecture 4). Also print the difference between \( x^* \) and the current estimate of the zero.

Comment on the behavior of the iterates in each case. Is there a pattern involving iterations when the Wheeler factor is reduced from 1? What happens to cause the Wheeler factor to be reset to 1? Explain, citing specific numbers in your results.

(d) Run fzero (see Exercise 2.1), defining \( x_0 \) as an interval specified by two initial points, with the same three pairs of points as in part (b): (i) \( x_0 = -0.6 \) and \( x_1 = 4 \); (ii) \( x_0 = 1 \) and \( x_1 = 5 \); (iii) \( x_0 = 5.6 \) and \( x_1 = 0.1 \) What information about the logic of fzero can you deduce by examining the iterates of fzero on this problem? For example, does fzero sometimes do bisection steps? Secant steps?

(e) Comment on the different features of the four methods applied to these problems, with particular emphasis on the role (if any) of the starting points, and on how well each method’s behavior in practice matched your expectations from the theory. Were there any surprises?