Math 4500/6500 Homework: Root Finding

This homework assignment covers our notes on solving equations. You are welcome to look at the code from the *Mathematica* notebooks, but when the problems say "write a piece of code to" they mean "write your own code from scratch", not "modify the code in the notebook" or "find a piece of code on the web".

1. (Warmup Bisection Proof) Write out a proof that if $f(a)f(b) < 0$ and $f(c) \neq 0$, then either ($f(a)f(c) < 0$ and $f(b)f(c) > 0$) or ($f(a)f(c) > 0$ and $f(b)f(c) < 0$).

2. (Bisection Proof) Suppose that $c_0, c_1, \ldots, c_n$ are the approximations produced by the bisection method to the solution r of the equation $f(x) = 0$ on an interval $[a_0, b_0]$. Prove that

$$|r - c_n| \leq \frac{b_0 - a_0}{2^{n+1}}.$$

3. (Newton Proof) Suppose that $f(x) = (x - t_*)(x - t_{**})$ where $t_{**} > t_* > 0$. Consider the iterates $x_0, x_1, \ldots, x_k, \ldots$ for Newton's method with starting point $x_0 = 0$.

   a. Prove that the iterates x_k form an increasing sequence, converging to t_*.

   b. Prove that the difference $t_* - x_k <= C(t_* - x_{k-1})^2$ for some fixed $C > 0$.

   c. Find a explicit constant C that works in the equation above.

4. (FindRoot) Mathematica has a built-in function called FindRoot which can be used to solve equations in the form $f(x) = 0$.

   a. Use FindRoot function to find a *nonzero* solution to the equation

$$6(e^x - x) = 6 + 3x^2 + 2x^3$$

   starting from $x = 2.0$ or $x = 3.0$ which is within 10^{-8} of the true solution. Use the PrecisionGoal option to specify the allowable error in the solution.

   b. Use the EvaluationMonitor option and Sow and Reap to collect a list of trial solutions $\{x_0, \ldots, x_n\}$ evaluated by FindRoot (see the documentation page for FindRoot for an example).

5. (The Bisection Method) Refer to the notes to refresh your memory of the bisection method for finding a root of $f(x)$ on an interval $[a, b]$ assuming that $f(a)f(b) < 0$.

   a. Write your own code to implement the bisection method in *Mathematica* for a general function $f(x)$ and endpoints a and b. Your function must:

      (1) be called `(lastname)Bisection[f,a,b,`ϵ`]`,

      (2) run the bisection method for an arbitrary function $f(x)$ on an interval $[a, b]$,

      (3) produce an error message and stop if $f(a)$ and $f(b)$ have the same sign,

      (4) terminate when the size of the current interval $[a_n, b_n]$ is less than ϵ

      (5) return a list in the form $\{\{a_n, b_n\}, \frac{a_n+b_n}{2}\}$.

b. Use your code to find a numerical solution of the equation

$$6(e^x - x) = 6 + 3x^2 + 2x^3.$$

starting with the interval $[2.0, 3.0]$ with absolute error less than 10^{-8}.

c. Compare your result to the result obtained from FindRoot in Problem 4.

6. (The False Position Method) The false position method is an alternative to the bisection method for solving the equation $f(x) = 0$ on an interval $[a, b]$ which works as follows. Given an interval $[a, b]$ and a continuous function $f(x)$ on the interval where $f(a)f(b) < 0$ (that is, $f(a)$ and $f(b)$ have different signs), construct a "false position" c for the point $x_* \in [a, b]$ by taking the point where the line joining $(a, f(a))$ and $(b, f(b))$ crosses the x-axis. The false position method then shrinks the interval to $[a, c]$ (if $f(a)f(c) < 0$) or $[c, b]$ (if $f(c)f(b) < 0$) and starts again, as in the bisection method. Unlike the bisection method, the false position method is guaranteed to move one end of the interval close to a root, but stops changing other end.

a. Write your own code to implement the false position method in *Mathematica*. Your function must:

(1) be called `(lastname)FalsePosition[f,a,b,ϵ]`,

(2) run the false position method for an arbitrary function $f(x)$ on an interval $[a, b]$,

(3) produce an error message and stop if $f(a)$ and $f(b)$ have the same sign,

(4) terminate when $|a_n - a_{n-1}|$ and $|b_n - b_{n-1}|$ are both less than ϵ

(5) return a list in the form $\{\{a_0, b_0, c_0\}, \ldots, \{a_n, b_n, c_n\}\}$ where c_k is the false position computed at the k-th step.

Note: If you're using NestWhileList to iterate, you'll be a little bit puzzled about how to write a test function which knows about *both* $[a_n, b_n]$ and $[a_{n-1}, b_{n-1}]$. One solution to the puzzle is in the documentation– the test function can get several iterates (or all the results so far) with optional arguments to NestWhileList.

b. Use your code to find a numerical solution of the equation

$$6(e^x - x) = 6 + 3x^2 + 2x^3.$$

starting with the interval $[2.0, 3.0]$ with absolute error less than 10^{-8}. Compare your result to the result obtained from FindRoot in Problem 4 and from the Bisection Method in Problem 5.

7. (Newton's Method in 1-d) Recall (or look up) the definition of Newton's method for scalar functions $f(x)$.

a. Write your own Mathematica code to implement Newton's method given the function and its derivative. Your function must:

(1) be called `(lastname)Newton[f,df,x0,n]`,

(2) run Newton's method for n steps for a function $f(x)$ with derivative $df(x)$ from the starting point $x0$,

(3) return a list in the form $\{\{x_0, f(x_0)\}, ..., \{x_n, f(x_n)\}\}$,

(4) produce an error message and stop if $df(x_k) = 0$ at any step k.

b. Use your code to find a numerical solution of the equation

$$6(e^x - x) = 6 + 3x^2 + 2x^3.$$

in the interval $[2.0, 3.0]$ from a starting point of your choice. Compare your result to the result obtained from FindRoot in Problem 4, from the False Position method of Problem 6, and from the Bisection Method of Problem 5.

8. (Compare and Contrast) You now have four different algorithms for finding roots implemented: the Bisection Method, the False Position Method, Newton's Method and FindRoot.

   a. Test them at finding solutions of $\tan x + \tanh x = 0$ and $x^2 - 22x + 3 = 0$ from several starting points and intervals, and write up your results, including tables of the intermediate values produced by each algorithm.

   b. Decide which method works best for each problem and justify your choice.

1. Challenge Problems

1. (Relative Bisection) Devise a modified bisection algorithm which guarantees that the result has *relative* error less than ϵ. Implement your method in *Mathematica*.

2. (Symbolic Differentiation) Write a Mathematica function which uses Mathematica to compute the derivative of the input function $f(x)$ for Newton's method. Your function must:
   (1) be called `(lastname)NewtonDiff[f,x0,n]`.
   (2) use Mathematica to compute $df(x)$,
   (3) run Newton's method for n steps for a function $f(x)$ with derivative $df(x)$ from the starting point $x0$.
   (4) return a list in the form $\{\{x_0, f(x_0)\}, ..., \{x_n, f(x_n)\}\}$.
   (5) produce an error message and stop if $df(x_k) = 0$ at any step k.
   This will require some noodling around with documentation in order to get it to work as you expect it to.

3. (Steffensen's Method) The **Steffensen method** is sometimes used in place of Newton's method when the derivative of $f(x)$ is known to exist, but it cannot be easily calculated. The Steffensen method has the iteration

$$x_{n+1} = x_n - \frac{f(x_n)}{g(x_n)} \quad \text{where } g(x) = \frac{f(x + f(x)) - f(x)}{f(x)}.$$

   (1) Implement the Steffensen method in *Mathematica*.
   (2) Find solutions of the equation $f(x) = (x-1)^8$ using both Newton's method and the Steffensen method.
   (3) Prove that when $f(x)$ is small, $g(x)$ is close to $f'(x)$ using Taylor series.
   (4) Prove that the Steffensen method is quadratically convergent.

4. (Newton-Kantorovich) Suppose that you are given a function $f(x)$, its derivative $df(x)$, its second derivative $ddf(x)$.

a. Use the Newton-Kantorovich theorem to find an estimated upper bound on the distance from the current iterate x_k to the nearby solution t_* when the NK conditions hold. You'll have to assume that the current value of $ddf(x)$ (or a multiple of it to be on the safe side) is an upper bound for all values of $ddf(x)$ nearby, but this usually isn't too far off.

b. Write a modified Newton's method code in Mathematica using this bound. Your function must
   (1) be called `(lastname)NewtonNK[f,df,ddf,x0,ϵ]`.
   (2) run Newton's method for a function $f(x)$ with derivative $df(x)$ from $x0$.
   (3) decide whether the Newton-Kantorovich conditions hold at each step, and compute an estimated upper bound $u_k > |x_k - x_*|$ if they do,
   (4) terminate when $u_k < \epsilon$,
   (5) return a list in the form
   $$\{x_0, f(x_0), df(x_0), ddf(x_0), u_0\},$$
   $$\dots$$
   $$\{x_n, f(x_n), df(x_n), ddf(x_n), u_n\}$$
   where u_k is the upper bound above if the Newton-Kantorovich conditions hold and ∞ if they don't,
   (6) produce an error message and stop if $df(x_k) = 0$ at any step k.

c. Test your method at finding solutions with error $< 10^{-8}$ to the problems $\tan x + \tanh x = 0$ and $x^2 - 22x + 3 = 0$.

5. (Iterative Division) Prove that for any $R > 0$ there is an interval of x_0 values around $1/R$ so that the limit of the iteration
$$x_{n+1} = x_n(2 - x_n R)$$
is $1/R$. This iteration was actually used to implement the division operation on some old computers which had circuitry for multiplication but not division!