Assignment 3-Written: Due 2/9

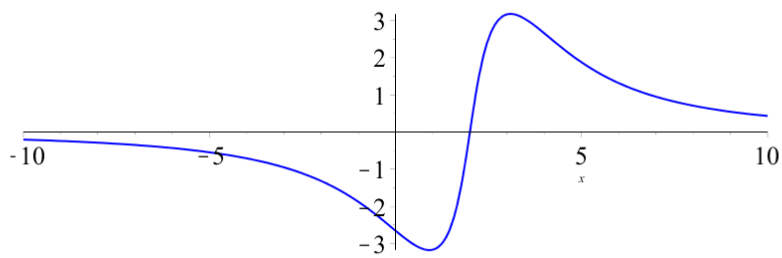
1) For the equation $x^3 - 3x = 1$; with initial bracketing interval $[0, 2]$, Perform 2 iterations of the bisection method. When completed, give the estimated root.

2) For $f(x) = e^x - 4x - 2$, and $x_0 = 3$; (a) Perform 2 iterations of Newton's method. (b) Find the backward error for x_0, x_1, and x_2. Do think you are making progress towards the root?

3) For $f(x) = x^2 - 6$, use the Secant Method with $x_0 = 3.1$, $x_1 = 3$; Specifically, Find x_2 and x_3.

4) Consider $f(x) = (x - 2)^1 3$; There is a root at 2. Which method would converge faster: Newton or Bisection?

5) Examine the graph of $f(x)$ below. There is a root near 2 (the only root!) Suppose you have know real idea where that root is. Explain the risks involved in using Newton's method to find the root.



Partial List of Answers:

1) Final bracket interval $[1.5, 2]$. Estimated root $x_{\text{app}} = 1.75$.

2) (a) $x_2 \approx 2.49133$; (b) $|f(x_0)| \approx 6.1$; $|f(x_1)| \approx 1.3$; $|f(x_2)| \approx 0.11$

3) $x_3 \approx 2.45536$

Assignment 3: Coding

Naming Convention: `yournameHybrid.m`, where your is the first 7 or less letters of your last name, followed by first initial. I would we `bieseckmHybrid.m`, while Sandra Poe would have `poesHybrid.m`

Write a Octave/Matlab function that is a hybrid of Bisection and Secant. The idea is attempt secant iterations unless we fall outside the bracketing interval. Then two bisection iterations are performed and the Secant is started again.

Input Arguments: Function f; bracket endpoints a, b; Tolerance `xTol`
Output Arguments: Approximate Root; Backwards Error

PseudoCode

```
 1) Evaluate Fcn:  fa=f(a); fb=f(b);  Check for bracket validity
     if fa*fb> 0,  then
        Tell user they are dumb.
        Return empty root:  root=[];
     elseif fa ==0
         return root=a
     elseif fb==0
         return root=b

2)  Set maxIters = log( abs(b-a)/xTol )/log(2)

3) Initialize Secants Variables:  x0=a; x1=b; f0=fa; f1=fb; doSecant=1;

4)  Loop 1 to maxITers
   4.1 if doSecant==1 then perform secant step
         x2=(x0*f1 - x1*f0)/(f1-f0)
         f2 = f(x2);
   4.2 Check validity of secant step
      if (x2>b) or (x2<a) or (f1==f2)  % Failed Iteration
           Set doSecant=-1;
      else get ready for another secant iteration
          x0=x1; f1=f0;
          x1=x2; f2=x2;
   4.3 if doSecant < 1  then bisect:
       m=0.5*(a+b); fm=f(m)
       if fa*fm >  0 then set a=m; fa=fm;
       if fb*fm >  0 then set b=m; fb=fm;
       if fm==0  then return root=m;
       set doSecant=doSecant+1;
       if doSecant==1 then we are ready for another secant try:
           set x0=a; x1=b;  f0=fa; f1=fb;
   4.4  Check Stopping Criteria  (if met return
        4.4.1:  if last step was bisection & if abs(b-a) is small
                     return  root = 0.5*(a+b);
        4.4.2:  if last step was successful secant & if (abs(x1-x0)/abs(x0)+1e-99) is small
                     return  root=x1;

5)  Optional:  if maxIters is reached, perform a fixed number of iterations of pure bisection
```