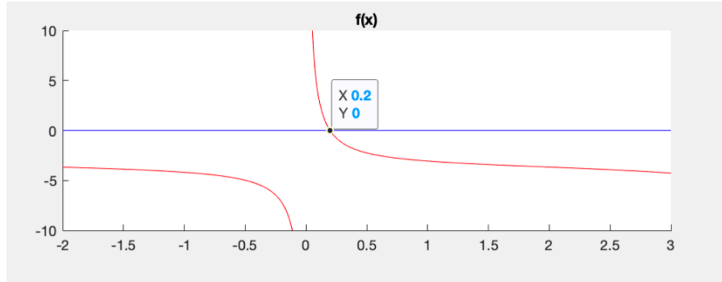
Minh Nguyen (301539625)
MACM 316

Computing Assignment 3 – Hybrid Newton-Bisection Algorithm

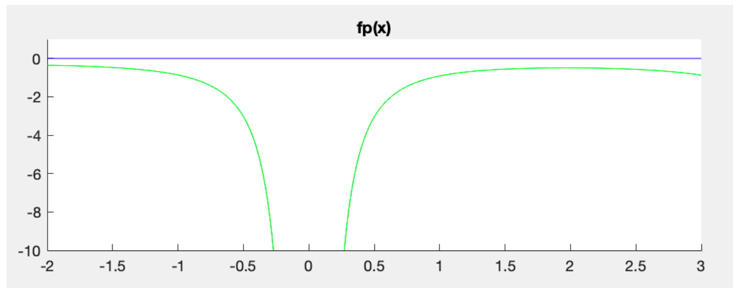
The purpose of this computing assignment is to combine Newton and Bisection method to create a hybrid method that is always convergent to the root with decent speed.

a) $f(x) = \frac{(1-x)*(3+x)^{\frac{1}{3}}}{x*(4-x)^{\frac{1}{2}}}$



$$f'(x) = -\frac{x^3 + 10x^2 - 11x + 72}{6(4-x)^{\frac{3}{2}}x^2(x+3)^{\frac{2}{3}}}$$

As we can observe the two plots of f(x) and f'(x), f(x) generally decreases with a steep curve on the interval [-2,3]. f(x) is not continuous as x approaches 0, f(x) goes to -∞ as x goes to 0^- and f(x) goes to ∞ as x goes to 0^+.



On the interval [-2,3], f(x) has only one root, which is approximated to be x≈0.2.

b)

Attempt to approximate the positive root of f(x) using Newton's method diverges due to the steep curve.

Bisection method will give a result of x ≈ 1.9802722930908201e-01 with 19 iterations for 10^{-6} tolerance.

c) With the given interval [0.1,1], the middle number x_0=0.55 will be obtained as the initial guess. However, Newton's method still diverges.

d) newtBrack function works as a filter to whether choose the next x obtained from Newton method or not. If the x obtained from the Newton method is out of the interval, the middle number of the interval will be chosen as the next x. If the x obtained from the Newton method is on the interval, it will be chosen as the next x.

e) The result from **newtonb** function:

Warning: xnewt is out of range.
xnewt is: -3.8481567166017039e-01, but the interval is [1.000000e-01,1].
With 8 iterations this is the root obtained from hybrid method: 0.198028306981218.
The list of x is:

0.550000000000000
0.325000000000000
0.118540187151740
0.166220003960436
0.192950164220103
0.197899155971562
0.198028223479739
0.198028306981218

Comment: Compare to Bisection method, the Hybrid method is faster even though it has a smaller tolerance, which is 10^{-10}.

The Hybrid is a robust method because it has advantages of both methods.

Minh Nguyen (301539625)
MACM 316

CA3.m | newtonb.m | newtBrack.m | newton.m

```matlab
%Minh Nguyen (301539625) %CA3-MACM 316
format long;
f=@(x) ((1-x).*(3+x).^(1/3))./(x.*sqrt(4-x))-3.06;
fp = @(x) -(x.^3 + 10*x.^2 - 11*x + 72)./(6*(4-x).^(3/2).*x.^2.*((x+3).^(2/3)));
tiledlayout(2,1);
nexttile;
hold on;
xint=-2:0.001:3;
axis([-2 3 -10 10]);
plot(xint, f(xint),'r');
title("f(x)");
yaxis=zeros(length(xint),1);
plot(xint,yaxis,'b');
hold off;
nexttile;
hold on;
plot(xint, fp(xint),'g');
title("fp(x)");
axis([-2 3 -10 10]);
plot(xint,yaxis,'b');
hold off;
%Newton's method
%[xNewton, nNewton, xlNewton] = newton( f, fp, 1, 10^(-6));
%display(xNewton);
%Bisection
[xB2,nB2,rlistB2]=bisect2(f,[0.1,1.0],10^(-6));
fprintf("With %d iterations this is the root obtained from Bisection" + ...
    " method: %.16d for 10^-6 tolerance.\n",nB2,xB2);
%Newtonb
[xNB, nNB, xlNB] = newtonb( f, fp, [0.1,1], 1e-10 );
fprintf("With %d iterations this is the root obtained from hybrid method:%.16d.\n",nNB,xNB);
display(xlNB);
%end here
```

CA3.m | newtonb.m | newtBrack.m | +

```matlab
function [ok,xnewt]=newtBrack(a,b,x,fx,fpx)
xnewt=x-fx/fpx;
ok=true;
if (xnewt<a || xnewt>b)
    ok=false;
end
end
```

CA3.m | newtonb.m | newtBrack.m | newton.m | +

```matlab
%Hybrid Newton & Bisection
function [root, iter, xlist] = newtonb( func, pfunc, xint, tol )
format long;
if nargin < 3
  fprintf(1, 'NEWTON_B: must be called with at least three arguments\n' );
  error( 'Usage:  [root, niter, xlist] = newton( func, pfunc, xint, [tol] )' );
end
if length(xint) ~= 2, error( 'Parameter ''xint'' must be a vector of length 2.' ), end
if nargin < 4, tol  = 1e-10; end
% fcnchk(...) converts function parameters to the correct type
% to allow evaluation by feval().
maxiter = 1000;  % don't iterate forever
func = fcnchk( func );
pfunc= fcnchk( pfunc );
xmid    = 0.5 * (xint(1) + xint(2));
x=xmid;
fx   = feval( func,  xmid );
fpx  = feval( pfunc, xmid );
if( fx == 0 || fpx == 0 )
  error( 'NEWTONB: both f and f'' must be non-zero at the initial guess' );
end
xlist= [ xmid ];
done = 0;
iter = 1; %The number of iterations is initialized as 1 because the first
% step using Bisection to find the initial guess is counted as the first iteration.
while( ~done )
  x0  = x;
  fx0  = feval( func,  x0 );
  fpx0 = feval( pfunc, x0 );
  [ok,xnewt]=newtBrack(xint(1),xint(2),x0,fx0,fpx0);
  x=xnewt;
  fx  = feval( func,  x );
  if (ok==false)
      fprintf("Warning: xnewt is out of range.\n");
      fprintf("xnewt is: %.16d, but the interval is [%d,%d].\n",xnewt, ...
          xint(1), xint(2));
      if fx0 * feval(func, xint(1)) < 0,
          xint(2) = xmid;
      else
          xint(1) = xmid;
      end
      xmid = 0.5 * (xint(1) + xint(2));
      x=xmid;
      fprintf("Thus, the xnewt is xmid %d.\n\n",xmid);

  end
  xlist = [ xlist; x ];   % add to the list of x-values
  iter  = iter + 1;
  if( iter>maxiter||abs(fx) < tol)
    done = 1;
  end
end

root = x;
%END newton.
```