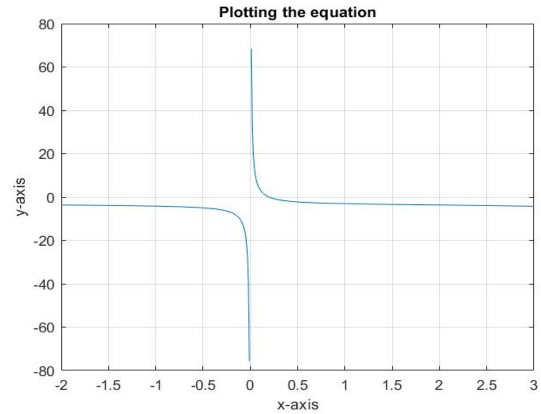
RITIKA GOYAL (301401516)

MACM316 CA3

The goal of this assignment is to find roots of equations by using hybrid approach which combines newton's and bisection method. An initial interval is passed as an argument in newton's method instead of passing initial guess. The mid-point of the interval is calculated and then it is used as an initial guess in newton's formula. After using newton's formula, the new found guess is then checked if it is in the given interval or not. If it is in the given interval, the normal newton formula is used to find another guess but if it is not, the bisection method is used (only on iteration) to calculate new guess and then it is again passed in newton's method. Bisection method take many iteration and newton does not guarantee to converge, so this hybrid approach is found to tackle both the issues.



Part a) By plotting the function f(x) = 3.06 - (((1-x) *(3+x) ^ (1/3))/ (x* ((4-x) ^ (1/2)))) on the interval [-2,3], the resulted plot is shown in this image. The graph has vertical asymptote at x = 0 and approaches infinity at x=0, it has one root at x = 0.19808415.

Part b) The roots are computed of the equation mentioned in part a by using bisection method, but newton's method diverges from the root at the initial guess (x=1) and thus, it runs infinitely:

<u>Bisection Method:</u>

Root = 0.198027 Initial Interval = [0.1,1.0] Number of Iteration = 19 (with 10^-6 tol) , 33 (with 10^-10 tol)

<u>Newton's Method:</u>

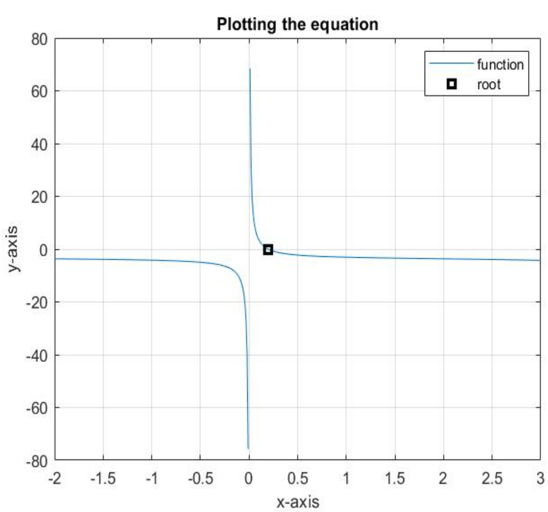
Root = no root as it diverges Initial Guess = 1 Number of Iteration = not calculated

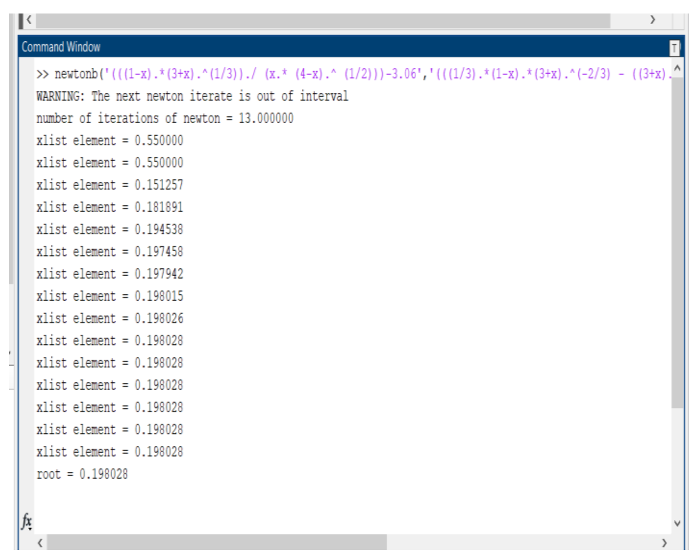
Part c) The newton's method is altered by adding initial interval instead of initial guess. The mid point of the interval is found by just using one bisection iteration and then the newton's formula is used to calculate the root. The calculated midpoint 0.55 is used as initial guess but it runs infinitely because at 0.5, divergence is observed.

Root = no root as it diverges Initial Interval = [0.1.1.0] Number of Iteration = not calculated

Part d) Implemented another function named newtBrack which takes interval values, x value, f(x) and fp(x) as an argument and return new value of x using newton formula and logical variable ok, which returns 1 if the new value is in the given interval and false otherwise. Then newton method is modified to use this newtonBrack function and if ok is false, warning message is displayed.

Part e) The newtonb.m shows the merging of bisection and newton method to calculate root. If the initial guess of newton method diverges, then bisection is used to calculate the new guess to get closer to root and then applied the newton on it by calling newtBrack. The new root is added in xlist and printed at end. The new plot with roots is displayed below with the output of the code representing all the elements in xlist in below table.





RITIKA GOYAL (301401516)

MATLAB CODE

```
%Part a
function myfunc()
x=-2:0.01:3;
y = 3.06-(((1-x).*(3+x).^(1/3))./ (x.* ((4-x).^ (1/2))));
plot(x,y); title("Plotting the equation");xlabel("x-axis");ylabel("y-axis");
grid on;

%Part b:
str_y = '(((1-x).*(3+x).^(1/3))./ (x.* (4-x).^ (1/2)))-3.06';
str_z = '(((1/3).*(1-x).*(3+x).^(-2/3) - ((3+x).^(1/3)) - ((4-x).^(1/2)) + ((1/2).*x.*(4-x).^(-
1/2)))./(x.^2.*(4-x)))';
[bisect_root, biter] = bisect2( str_y, [0.1,1.0] );
fprintf("bisection root = %f, niter = %f\n",bisect_root,biter);
[newton_root, niter] = newton( str_y, str_z, 1 );
fprintf("newton root = %f, niter = %f\n ",newton_root,niter);
end

%Part d)
function [ok, xnewt] = newtBrack(a, b, x, fx, fpx)
xnewt = x-(fx/fpx);
if(xnewt>=a && xnewt<=b)
    ok = 1;
else
    ok =0;
end
end

%part c) and part e)
function [root, iter, xlist] = newton( func, pfunc, xint, tol )
if nargin < 4, tol  = 1e-10; end
func = fcnchk( func );
pfunc= fcnchk( pfunc );
iter=0; xmid = 0.5 * (xint(1) + xint(2));
fmid = feval(func, xmid);
x0    = xmid;
fx   = feval( func,  x0 );
fpx  = feval( pfunc, x0 );
done = 0; iter = 0;
xlist= [ x0 ];
while( ~done )
  [ok,x]= newtBrack(xint(1),xint(2),x0,fx,fpx);
  if(ok == 0)
      fprintf("WARNING: The next newton iterate is out of interval\n");
      fmid = feval(func, x0);
      if( fmid * feval(func, xint(1)) < 0)
        xint(2) = x0;
      else
        xint(1) = x0;
      end
      xlist = [xlist; x0];
      x0=0.5 * (xint(1) + xint(2));
      fx  = feval( func,  x0 );
      fpx = feval( pfunc, x0 );
      if (abs(xint(2)-xint(1)) < 2*tol || abs(fmid) < tol)
        done = 1;
      end
  else
      fx  = feval( func,  x );
      fpx = feval( pfunc, x );
      if( abs(x-x0) < tol )     % absolute tolerance on x
        done = 1;
      else
        xlist = [ xlist; x ];   % add to the list of x-values
        iter  = iter + 1;
      end
      x0=x;
  end
end
fprintf('number of iterations of newton = %f\n',iter);
fprintf('xlist element = %f\n',xlist);
root = x;
x=-2:0.01:3; y = 3.06-(((1-x).*(3+x).^(1/3))./ (x.* ((4-x).^ (1/2))));
plot(x,y);legend('function');title("Plotting the equation");xlabel("x-axis");ylabel("y-axis");grid on;
hold on; plot(root,0,'sk','LineWidth',2,'DisplayName','root'); hold off; end
```