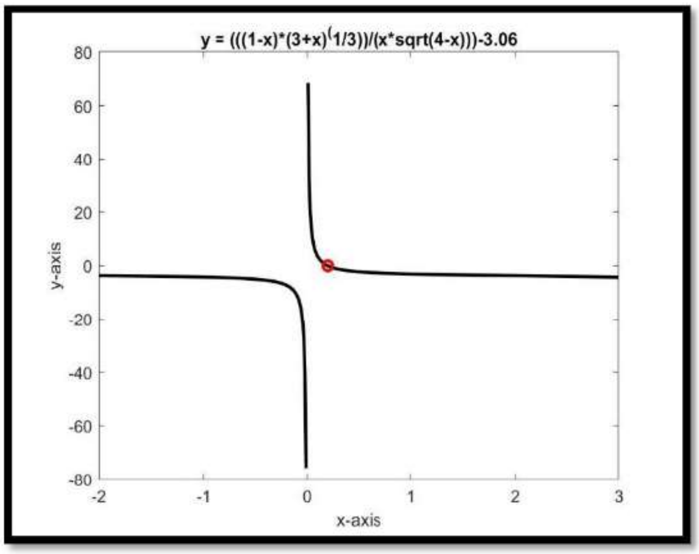
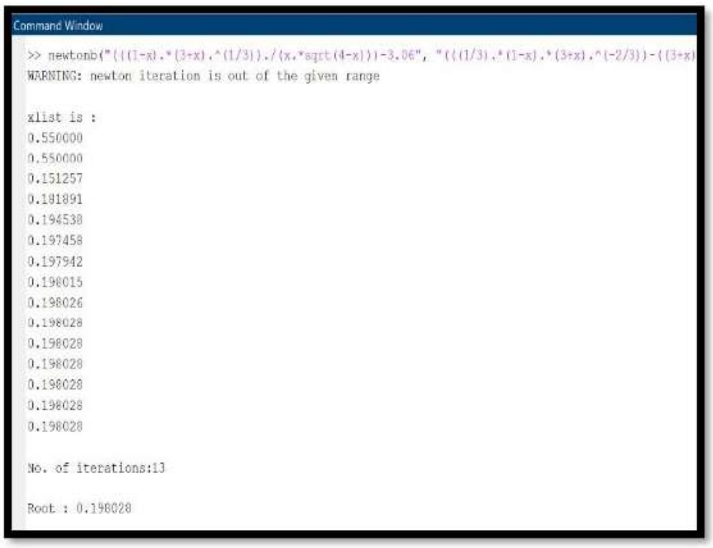
COMPUTING ASSIGNMENT 3

NAVJOT KAUR (301404765)

This computing assignment concerns with solving the equation with a hybrid method that combines Bisection and Newton method together because bisection method takes more iterations to converge and Newton's method is not guaranteed to converge. In order to use less iterations hybrid method is used. The equation to be solved is $f(x) = (\frac{(1-x)*(3+x)^{\frac{1}{3}}}{x*(4-x)^{\frac{1}{2}}} - 3.06.$




Command Window

```
>> newtonb("(((1-x).*(3+x).^(1/3))./(x.*sqrt(4-x)))-3.06", "(((1/3).*(1-x).*(3+x).^(-2/3))-((3+x)
WARNING: newton iteration is out of the given range

xlist is :
0.550000
0.550000
0.151257
0.181891
0.194538
0.197458
0.197942
0.198015
0.198026
0.198028
0.198028
0.198028
0.198028
0.198028
0.198028

No. of iterations:13

Root : 0.198028
```

In part a, the above given function is plotted in the interval [-2,3] where the equation has a vertical asymptote at x=0 as can be clearly seen from the graph. It is evident from the graph that this equation has one root because the plot crosses x-axis at one point only. By zooming in, the approximate value of the root is 0.198084.

In part b, when the above equation is solved by bisection method it converges to 0.198028 with 33 iterations, when $tol = 10^{-10}$ and xint = [0.1, 1.0]. However, the Newton's method diverges at initial guess x0 = 1, when used for this equation.

Since the Newton's method diverges, in order to solve the given non-linear equation, there is a need to introduce a hybrid method that takes less iterations than bisection method and converges to a root.

In part c, a MATLAB code(newtonb.m) is written for a new algorithm called newton bisection algorithm that uses one step bisection method to determine mid-point, xmid and then Newton's method is used to find a root using the initial guess, xmid. The new algorithm takes 4 arguments i.e, $f(x)$, $f'(x)$, x_interval and tol, where $tol = 10^{-10}$ and x_interval = [0.1, 1.0]. It can be noted that this method uses an interval for x rather than an initial guess, it is done so to make the method more reliable. Here, also this method diverges because when initial guess, xmid=0.550000, is used then the newton method diverges as can be interpreted from the plot.

In part d, the newtonb.m is modified that checks if the guess used for newton's method is within the specified range or not. This is done via new function called newtBrack.m, which returns the new guess and a logical variable "ok". Here when ok is false, it prints a warning and then another bisection iteration is performed so that the guess lies within the range.

In part e, the iterations are updated and a screenshot of xlist is provided which states that the function converges to 0.198028 in 13 iterations. Here the root = 0.198028 is plotted on the graph.

MATLAB CODE

```
function parta_b()
x=-2:0.01:3;                                                            %part a
y = (((1-x).*(3+x).^(1/3))./(x.*sqrt(4-x)))-3.06;
plot(x,y, "k", "LineWidth", 2); hold on;
xlabel("x-axis"); ylabel("y-axis"); title("y = (((1-x)*(3+x)^(1/3))/(x*sqrt(4-x)))-3.06");
func="(((1-x).*(3+x).^(1/3))./(x.*sqrt(4-x)))-3.06";                    %part (b)
pfunc="(((1/3).*(1-x).*(3+x).^(-2/3))-((3+x).^(1/3))-sqrt(4-x)+((x/2).*(4-x).^(-1/2)))./(4.*x.^2 -
x.^3)";
xguess=1; tol=1e-10;
[root, iter, xlist] = newton(func, pfunc, xguess, tol);
printf("Using Newton method\nRoot: %f\n Iterations: %d\n", root , iter);
xint=[0.1,1.0];
[root, niter, rlist] = bisect2( func, xint, tol );
fprintf("Using bisection method\nRoot: %f\nIterations: %d\n", root , niter);
[root]=newtonb(func, pfunc, [0.1,1.0] , tol);                          %plotting the root
plot(root, 0,"ro", "LineWidth", 2); hold off
end
function [ok, xnewt] = newtBrack(a, b, x, fx, fpx)                      %part d (newtBrack.m)
xnewt = x - (fx / fpx);
if(xnewt>=a && xnewt<=b)
    ok=1;
else
    ok=0;
end
end
function [root, iter, xlist] = newtonb( func, pfunc, x_interval, tol )   %part c, d, e (newtonb.m)
if nargin < 3
    fprintf(1, 'NEWTON_BISECTION: must be called with at least three arguments' );
    error( 'Usage:  [root, niter, xlist] = newtonb( func, pfunc, x_interval, [tol] )' );
end
if nargin < 4, tol  = 1e-10; end
func = fcnchk( func ); pfunc= fcnchk( pfunc );
xmid = 0.5 * (x_interval(1) + x_interval(2));
fmid = feval(func, xmid); x    = xmid;
fx   = feval( func,  x ); fpx  = feval( pfunc, x );
if( fx == 0 || fpx == 0 )
    error( 'NEWTON_BISECTION: both f and f'' must be non-zero at the initial guess' );
end
xlist= [ x ]; done = 0; iter = 0; x0  = x;
while( ~done )
    [ok, x] = newtBrack(x_interval(1), x_interval(2), x0, fx, fpx);
    if(ok==1)
        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;
    else
        fprintf("WARNING: newton iteration is out of the given range\n\n");
        fmid = feval(func, x0);
        if(fmid * feval(func, x_interval(1)) < 0)
            x_interval(2) = x0;
        else
            x_interval(1) = x0;
        end
        xlist = [xlist; x0];
        x0 = 0.5 * (x_interval(1) + x_interval(2));
        fx  = feval( func,  x0 );
        fpx = feval( pfunc, x0 );
        if (abs(x_interval(2)-x_interval(1)) < 2*tol || abs(fmid) < tol)
            done = 1;
        end
    end
end
disp("xlist is : ");
fprintf("%f\n", xlist);
fprintf("\nNo. of iterations:%d\n", iter);
root = x; fprintf("\nRoot : %f\n", root); end
```