Table of Contents

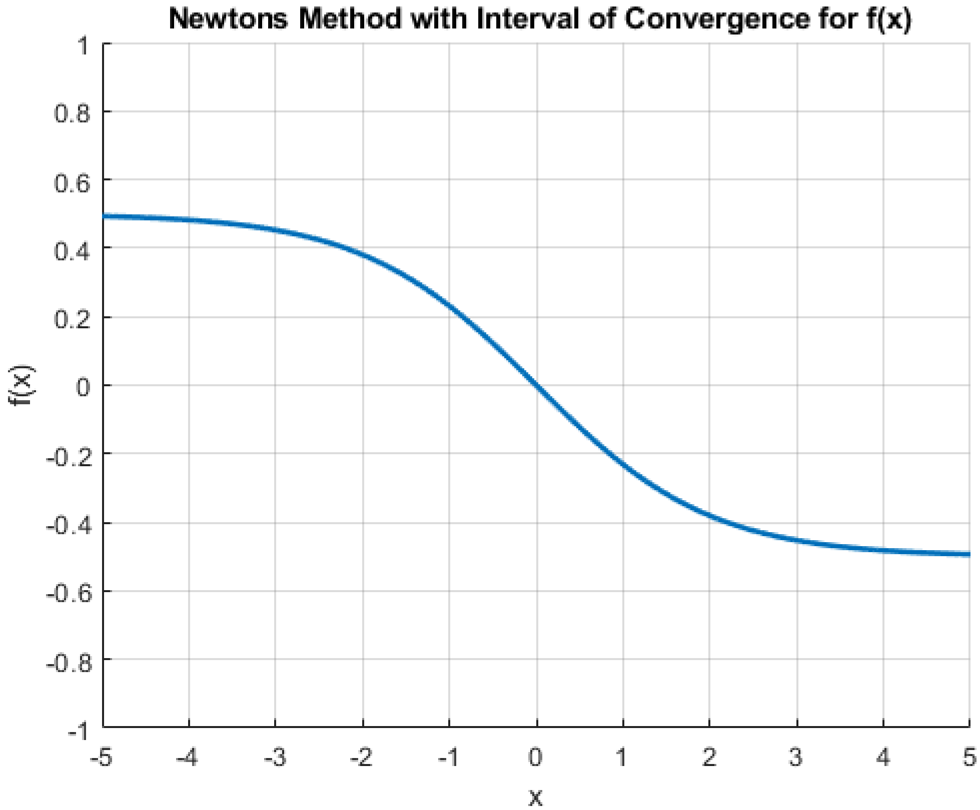
Housekeeping

```
clc; clear all; close all;
%{
    CSCI 3656 HW 3 main script
    Author: Connor O'Reilly
    Last edited9/16/2021
    SID: 107054811
%}
```

Step 1

Given f, its derivative f1 and interval to check x e [-5 , 5]

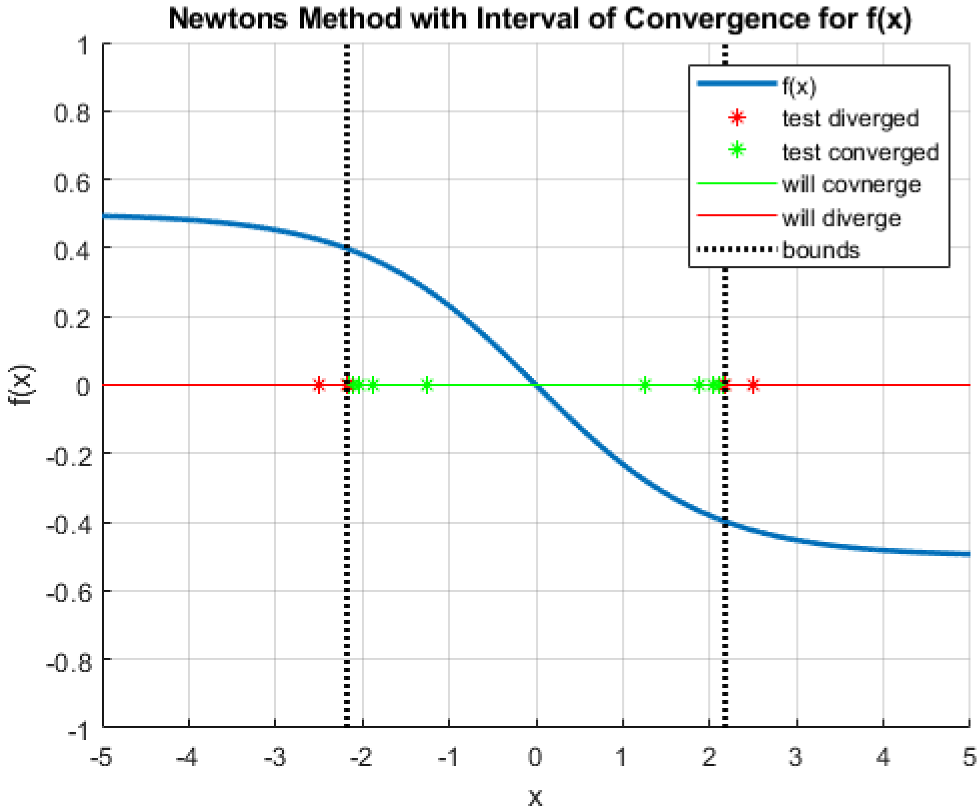
```
f = @(x) 1./(1+exp(x)) - (1/2);
f1 = @(x) -exp(x)./(1+exp(x))^2;
int = linspace(-5,5,1000);
%tolerance same as used in class
tol = 1e-9;
%plot func prior to convergence test
figure(1)
hold on;
grid on;
axis([-5 ,5, -1, 1])
title(' Newtons Method with Interval of Convergence for f(x)')
xlabel('x')
ylabel('f(x)')
plot(int,f(int),'Linewidth',2)
```



Steps 2-3

positive side of f(x) interval

```
b = newts_conv(f,f1,0,5,1e-9,100);
%negative side of f(x) interval
a = newts_conv(f,f1,0,-5,1e-9,100);
%no idea but error was introduced while messing with the plotting
 functions
%where a is out of bounds rounds towards posinfinity at 10000th
legend('f(x)' , 'test diverged' , 'test converged' , '' ,'' , ''
 , '' ,'' ,'' ,'' ,'' ,'' ,'' ,'' ,'' , '' ,'' , '' , '' ,'' ,''
 ,'' ,'' ,'' ,'' ,'' ,'' , '' ,'' , '' , '' ,'' ,'' ,'' ,'' ,'' ,''
 ,'' ,'',  '' ,'' , '' , '' ,'' ,'' ,'' ,'' ,'' ,'' ,'' ,'' , '' ,''
 , '' , '' ,'' ,'' ,'' ,'' ,'' ,'' ,'' ,'', '' ,'','','','', 'will
 covnerge' , 'will diverge' ,'bounds' )
a = ceil(a*10^8)/(10^8);
savefig('interval_newt.fig');
```



Step 4

```
fprintf('Length of b-a: %0.4f \n',b-a);
fprintf('Newtons method converges for any initial guess in the
 interval  [%0.8f , %0.8f]\n',a,b);
```

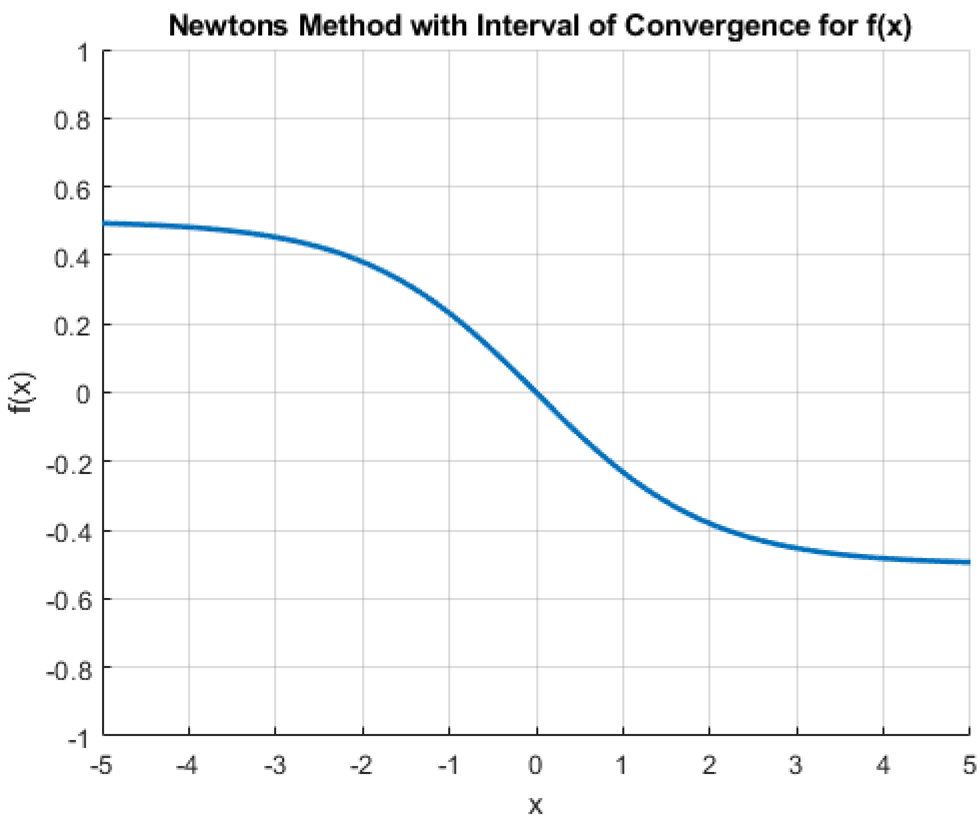
```
Length of b-a: 4.3546
Newtons method converges for any initial guess in the interval
 [-2.17731898 , 2.17731899]
```

Extra step

```
%test interval bounds
close all;
figure
title(' Newtons Method with Interval of Convergence for f(x)')
hold on;
grid on;
axis([-5 ,5, -1, 1])
xlabel('x')
ylabel('f(x)')
plot(int,f(int),'Linewidth',2)
 [approxa,~, itera] = newtons(f, f1, a, 1e-9, 200, true);
[approxb, ~, iterb] = newtons(f, f1, a, 1e-9, 200, true);
```

```
fprintf('Approximation of root and iteration count for left side
 limit, root = %0.8f, iterations: %d',approxa, itera)
fprintf('Approximation of root and iteration count for right side
 limit, root = %0.8f, iterations: %d',approxb, iterb)
```

```
Approximation of root and iteration count for left side limit, root =
 0.00000000, iterations: 20Approximation of root and iteration count
 for right side limit, root = 0.00000000, iterations: 20
```



Functions

```
function[a_or_b] = newts_conv(f,f1,a, b, tol, max_it)
%{
    used to determine the interval on which newtons method converges

    will iteratively check to determine the largest or smallest if
 interval
    is negative on which newtons methd converges following similar to
 bisection
%}
    %tolerance check
    a0 = a;
    b0 = b;
    i = 0;
    while (abs(b - a)/2 > tol )
        %tolerance check and make sure a non divergence value returns
```

```
        if i > max_it
            break;
        end
        %declare midpoint
        newmid = (a + b)/2;
        %determine if method converged or diverged at midpoint
        [~, did_it_bool,~] = newtons(f , f1 , newmid , tol,
 max_it,false);
        if did_it_bool
            plot(newmid,0,'g*');
        else
            plot(newmid,0,'r*')
        end
        %if method converged at midpoint, set left end to midpoint and
 go
        %again
        if did_it_bool
            a = newmid;
        else
            %if method converged at midpoint, set left end to midpoint
 and go
            %again
            b = newmid;
        end
        i = i + 1;
    end
    %return approximation of interval limit
    a_or_b = newmid;
    %add conv test to plot
    cool = linspace(a0,newmid,100);
    plot(cool,zeros(1,length(cool)),'g-');
    cool = linspace(newmid, b0,100);
    plot(cool,zeros(1,length(cool)),'r-');
    ylim = get(gca,'YLim');
    line([newmid newmid],
 ylim,'LineStyle',':','Color','k','LineWidth',2);

end
function [x, convbool, i] = newtons(f , f1 ,  x0 ,tol, max_it,
 coolplot)
%{
    MATLAB implementation of Newton's method
    Inputs:
        f: function to be evaluated
        f1: first derivative of function
        x0: Initial guess
        tol: tolerance
        max_it: maximum iteration count
        coolplot: boolean value which toggles the animation

    output:
        x: approximation of root
        convbool: returns if method covnerged
        i: number of iterations
```

```
%}
    %initialize parameters
    convbool = true;
    i = 1;
    x = x0;
    %values for plot
    %random interval for tangent line
    x_plot = linspace(-5,5,100);

    while  (max_it > i) && ( abs(subs(f,x)) > tol )

        x1 = x;
        x = x1 - f(x1)./f1(x1);
        i = i + 1;
        %plotting

        x1p = plot(x1, f(x1), 'ro', 'Linewidth', 1.5); %tangent point
        hold on;
        %for tangent line
        tline = f1(x1) .* (x_plot - x1) + f(x1);
        t = plot(x_plot, tline ,'b--','Linewidth',1.5);
        %second x point
        xp = plot(x,0,'go', 'Linewidth', 2);
        %divergence test ( will only work for given function i
 believe )
       vert =  plot([x x], [0 f(x)],'g--','Linewidth',1);
        if  abs(x) > abs(x1)
            %diverging
            x = NaN;
            convbool = false;
            break;
        end
        %show iteration live
        if coolplot
        pause(1);
        end
        delete(vert);delete(t); delete(xp); delete(x1p);
    end
    delete(vert);delete(t); delete(xp); delete(x1p);
end
```

Published with MATLAB® R2021a