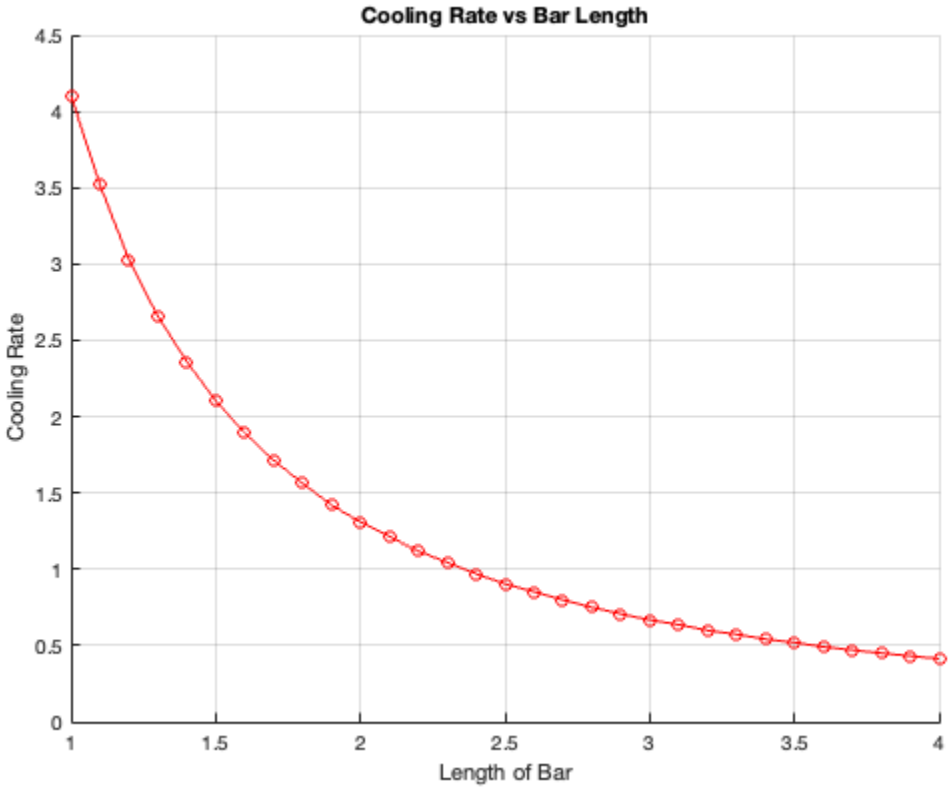
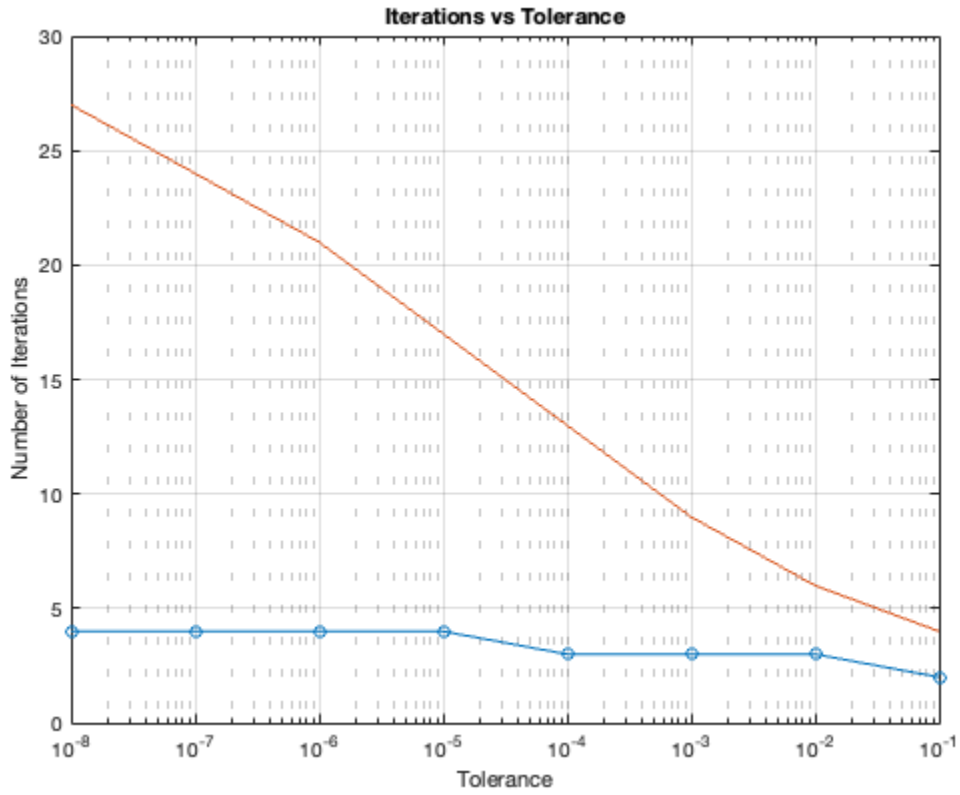
```
%Kaitlyn Kirt, CMOR 220, Spring 2024, Finding Real Roots Project
%Project2.m
%This script is a project on finding roots for one real variable functions
%Last motified: February 5, 2024

%driver
function Project2
%Bisection Method
disp("Bisection Method")
Bisection

%Newton's Method
disp("Newton Method")
Newton
end

function Bisection
f=@(x,L) sin(x*L)+x*cos(x*L);%anonymous function for given equation
L=1:0.1:4; %range of L values
x=zeros(1,length(L)); %preallocates x
a=0.1;
tol=0.01;
    for n=1:length(x) %runs code for "length of array x" times
        b=3/L(n); %defines b
        [x(n),~]=Bis(a,b,tol,L(n),f); %runs bisection with varying L
    end

figure(1);
hold on;
grid on;
plot(L,x.^2,"ro-")
title('Cooling Rate vs Bar Length')
xlabel('Length of Bar')
ylabel('Cooling Rate')
end

function [x,iter]=Bis(a,b,tol,L,f)
%inputs: a,b,tol,L,f
%outputs: x,iter
%description: this function uses the intermediate value theorem to cut the
interval in half to find a root
x=(a+b)/2; %defines the midpoint
iter=0;
while abs(f(x,L))>tol %assumes the absolute value of f(x,L) is greater than
0.01
    if f(a,L)*f(x,L)<0 %assumes f(a,L) and f(x,L) have opposite signs
        b=x; %label b as x
    else
        a=x; %set a as x
    end
    x=(a+b)/2; %restates x equation
    iter=iter+1; %adds 1 to iter then redefines iter
```

```
end
end

%Newton's Method
function Newton
f=@(x,L) sin(x*L)+x*cos(x*L); %creates anonymous function for given equation
dfdx=@(x,L) L*cos(x*L)-x*L*sin(x*L)+cos(x*L); %creates an anoynmous function
for derivative of f
a=0.1;
b=3;
x=(a+b)/2; %defines the midpoint
L=1;
tol=10.^-(1:8);
iteration=zeros(1,length(tol)); %preallocates interation
iteration1=zeros(1,length(tol)); %preallocates iteration1
    for n=1:length(tol) %runs code for "length of tol" times
        [~,iteration1(n)]=Bis(a,b,tol(n),L,f); %runs newton with varying tol
        [~,iteration(n)]=Newt(x,tol(n),L,f,dfdx); % runs newton with varying
tol
    end
figure(2);
semilogx(tol,iteration,"Marker","o")
hold on;
grid on;
semilogx(tol,iteration1)
title('Iterations vs Tolerance')
xlabel('Tolerance')
ylabel('Number of Iterations')
end

function [x,iter]=Newt(x,tol,L,f,dfdx)
%inputs: x,tol,L,f,dfdx
%ouputs: x,iter
%description: this function is a linear approximation to solve a root
iter=0;
    while abs(f(x,L))>tol %assumes absolute value of f(x,L) is greater than
10^-(1:8)
        x=x-f(x,L)/dfdx(x,L); %redefines x using Newton's mechanism of
tangent lines
        iter=iter+1; %adds 1 to iter then redefines iter
    end
end
```

Bisection Method
Newton Method


Cooling Rate vs Bar Length
Cooling Rate
Length of Bar
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1
1.5
2
2.5
3
3.5
4


Iterations vs Tolerance
Number of Iterations
Tolerance
30
25
20
15
10
5
0
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}

Published with MATLAB® R2023b