Table of Contents


```
% Joshua Oates  - Aero300 - Lab 3
```

Section 0 - cleanup

```
clear all;
close all;
clc;
```

Section 1 - test each 10,000 times

```
% create: f,fp,fpp,TOL,x0, stepSize, numRoots, searchDomain
f = @(x) x^3 + 1.0142*x^2 - 19.3629*x + 15.8398;
fp= @(x) 3*x^2 + 2.0284*x - 19.3629;
fpp=@(x) 6*x + 2.0284;

TOL=.5e-6;
x0 = 0;
stepSize = .1;
numRoots = 3;

brackets = JoshBracket(f,x0, stepSize, numRoots);
clear x0 % x0 will be used in other functions so I clear it for clarity
numBrakets = size(brackets);
numBrakets = numBrakets(1);

maxI = 10000; % large number to show shortcomings if a root is not found
 quickly
numTestIterations = 10000;

biTime = zeros(1,numTestIterations);
newTime = zeros(1,numTestIterations);
hallTime = zeros(1,numTestIterations);

biOut = zeros(1,numBrakets);
newOut = zeros(1,numBrakets);
hallOut = zeros(1,numBrakets);

% test bisection
for j = 1:1:numTestIterations
    tic
    for i = 1:1:numBrakets
        a = brackets(i,1);
```

```
        b = brackets(i,2);
        x0 = (a+b)/2;
        biOut(i) = JoshBisection(f,[a,b],TOL);
    end
biTime(j) = toc;
end

% test newtons
for j = 1:1:numTestIterations
    tic
    for i = 1:1:numBrakets
        a = brackets(i,1);
        b = brackets(i,2);
        x0 = (a+b)/2;
        newOut(i) = JoshNewtons(f, fp, x0, TOL, maxI);
    end
newTime(j) =  toc;
end

% test halleys
for j = 1:1:numTestIterations
    tic
    for i = 1:1:numBrakets
        a = brackets(i,1);
        b = brackets(i,2);
        x0 = (a+b)/2;
        hallOut(i) =JoshHalleys(f, fp,fpp, x0, TOL, maxI);
    end
hallTime(j) =  toc;
end

biSumTime = 0;
newSumTime = 0;
hallSumTime = 0;
for i = 1:1:numTestIterations
    biSumTime = biSumTime +  biTime(i);
    newSumTime = newSumTime + newTime(i);
    hallSumTime = hallSumTime + hallTime(i);
end

biAvTime = biSumTime/numTestIterations;
newAvTime = newSumTime/numTestIterations;
hallAvTime = hallSumTime/numTestIterations;
% clear vars that wont be used again
clear i j stepSize x0 a b

% print the results
disp("Bisection ran  "+ numTestIterations +" iterations in "+ biSumTime+" s,
 with an average time of "+biAvTime+" s")
disp("Newtons ran    "+ numTestIterations +" iterations in "+newSumTime+" s,
 with an average time of "+newAvTime+" s")
disp("Halleys ran    "+ numTestIterations +" iterations in "+hallSumTime+" s,
 with an average time of "+hallAvTime+" s")
```

```
format long
disp(" ")
disp("Bisection found the roots ")
for i =1:1:numBrakets
    disp(biOut(i))
end

disp("Newtons found the roots ")
for i =1:1:numBrakets
    disp(newOut(i))
end

disp("Halleys found the roots ")
for i =1:1:numBrakets
    disp(hallOut(i))
end

% discuss the results
disp("Bisection has the longest run time by nearly an order of magnitude. This
 makes sense beacause it is known that Bisection is much less effecient than
 other methods.")
disp("Newtons and Halleys have similar run times except that Halleys is
 slightly faster. This makes sense since Halleys method is similar to Newtons
 but involves slightly more information on the function")

% clear out vars
clear hallOut hallSumTime hallAvTime hallTime newOut newSumTime newAvTime newTime biOut bi
```

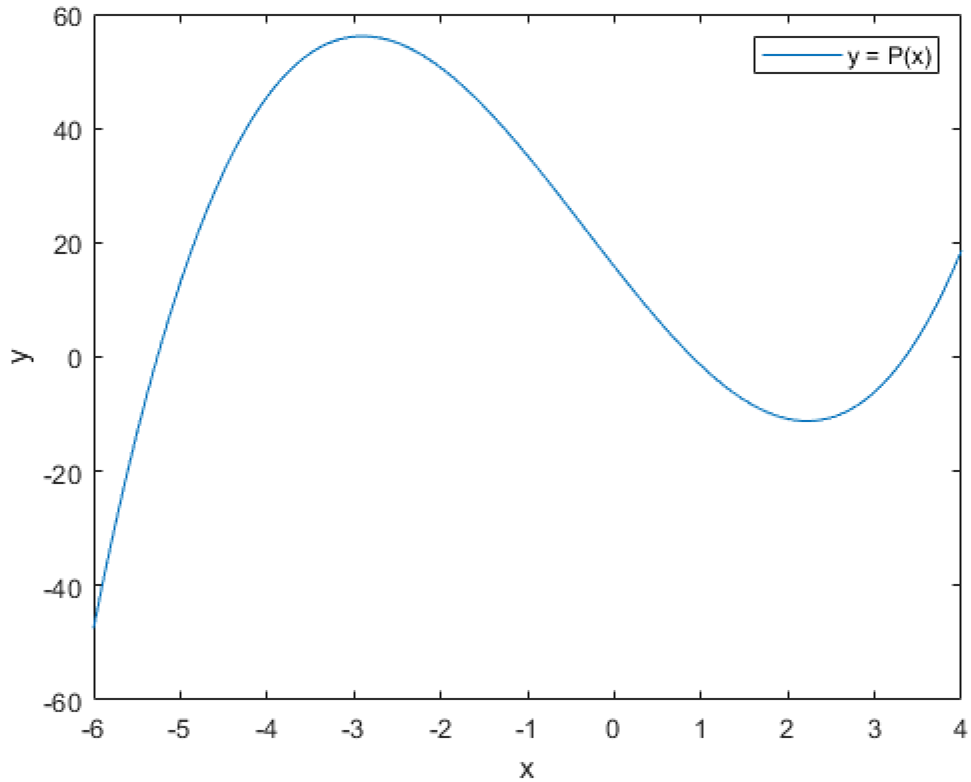
```
Bisection ran  10000 iterations in 0.55692 s, with an average time of
 5.5692e-05 s
Newtons ran    10000 iterations in 0.067316 s, with an average time of
 6.7316e-06 s
Halleys ran    10000 iterations in 0.058381 s, with an average time of
 5.8381e-06 s

Bisection found the roots
  -5.264102554321283

   0.897597885131842

   3.352304458618172

Newtons found the roots
  -5.264102458433700

   0.897598082398079

   3.352304376035620

Halleys found the roots
  -5.264102458433700

   0.897598082398080
```

```
   3.3523043760356200
```

Bisection has the longest run time by nearly an order of magnitude. This makes sense beacause it is known that Bisection is much less effecient than other methods.
Newtons and Halleys have similar run times except that Halleys is slightly faster. This makes sense since Halleys method is similar to Newtons but involves slightly more information on the function

Section 2 - Figure 1 – Plot and label the function p(x) from part 5

```
X = linspace(-6,4);
i=1;
for x = X
    Y(i) = f(x);
    i = i+1;
end

clear i x

figure('name',"P(x) visualization")
plot(X,Y)

xlabel("x")
ylabel("y")
legend("y = P(x)")
```

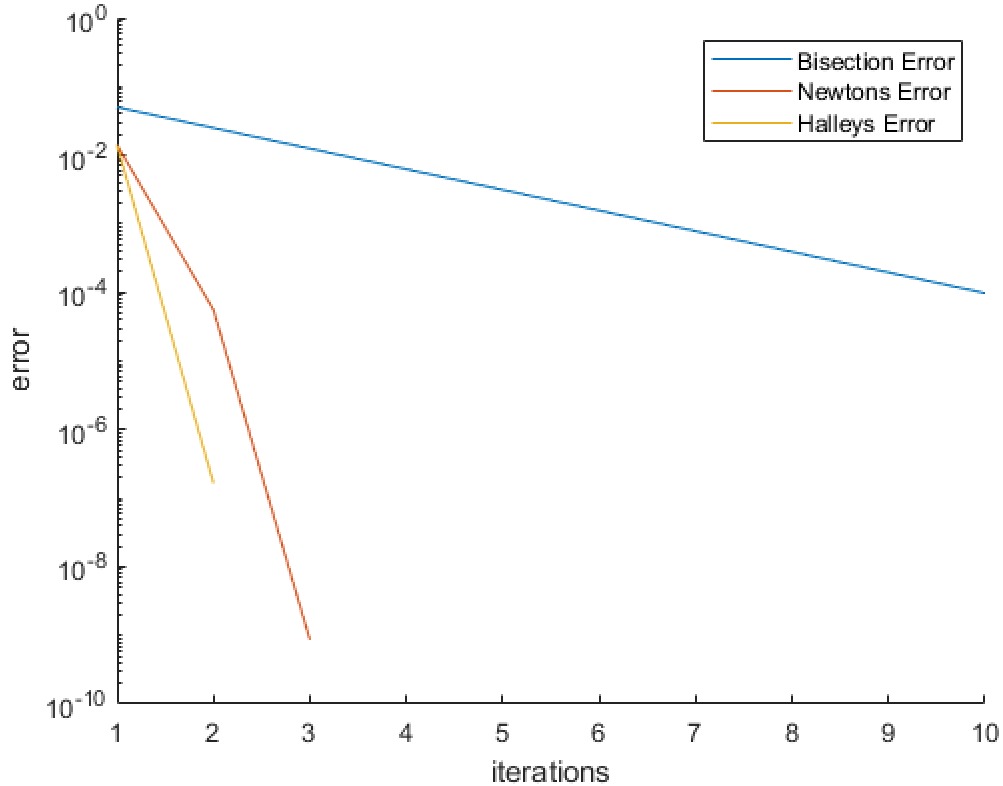


Section 3 - semilog() and label the iteration number versus the absolute error

```
a = brackets(1,1);
b = brackets(1,2);
x0 = (a+b)/2;
EVB = []; % Error Value Bi


for i=1:1:10
    EVB = [EVB,(b-a)/2^i] ;
end

TOL = 0;
[~,~,~,EVN] = JoshNewtons(f, fp, x0, TOL); % Error Value Newtons
[~,~,~,EVH] =JoshHalleys(f, fp,fpp, x0, TOL); % Error Value Halleys

figure('name',"comparison of error against iteration")
hold on
i=1:1:length(EVB);
plot(i,EVB)
polyB = polyfit(i,EVB,1);

i=1:1:length(EVN);
```

```
plot(i,EVN)
polyN = polyfit(i,EVN,1);

i=1:1:length(EVH);
plot(i,EVH)
polyH = polyfit(i,EVH,1);

set(gca,'yscale',"log")
xlabel("iterations")
ylabel("error")
legend("Bisection Error","Newtons Error","Halleys Error")
disp("the linear slope of the bisection error is ")
disp(polyB(1))
disp("the linear slope of the Newtons error is ")
disp(polyN(1))
disp("the linear slope of the Halleys error is ")
disp(polyH(1))
disp("the steepest lines have higher convergence while the shallower have
 lower. This makes sense since Bisection is known to have a much lower
 convergence than Halleys or Newtons")
```

```
the linear slope of the bisection error is
  -0.004250118371212

the linear slope of the Newtons error is
  -0.004253012994344

the linear slope of the Halleys error is
  -0.007051146317518

the steepest lines have higher convergence while the shallower have lower.
 This makes sense since Bisection is known to have a much lower convergence
 than Halleys or Newtons
```



Published with MATLAB® R2022a