Table of Contents


```
clear
clc
```

Problem 1

```
tol=1e-6;
max_its=100;
fcn='x^5-x^4+x-1';
trueroot=1;
intervals=[0, 3; 0.5, 2.0; 0.9, 1.2];

disp(sprintf('\nBisection estimate for root of %s with accuracy of
 %g:',fcn,tol));
disp(sprintf('_interval_ \t _estimate_ \t _error_ \t _iterations_'))

for i=1:size(intervals,1),

 [itiB(i),rootiB(i)]=bisect(fcn,intervals(i,1),intervals(i,2),tol,max_its);
  disp(sprintf('[%g,%g]    \t %0.8f \t %0.5e \t %d',...
        intervals(i,:),rootiB(i),abs(trueroot-rootiB(i)),itiB(i)));
end
% a. As it starts at 0.5 which makes the size of initial interval
 small to
% make the midpoint closer to true root

% b. yes, it is better to have the root closer to the midpoint of the
 interval
% as bisection algorithm splits the interval into two parts, then it
 will evaluate
% the function by using those two domains, so when the domain of the
% interval becomes and the midpoint becomes closer to the root, the
 number of iterations becomes
% less due to that. For example, the interval [0,3], which its
 midpoint is 1.5, with 21 iterations
% which is larger than the interval [0.9, 1.2], which its midpoint is
 1.05, with 18 iterations
```

```
Bisection estimate for root of x^5-x^4+x-1 with accuracy of 1e-06:
_interval_   _estimate_   _error_   _iterations_
```

```
[0,3]       0.99999976   2.38419e-07   21
[0.5,2]       1.00000024   2.38419e-07   20
[0.9,1.2]       1.00000019   1.90735e-07   18
```

Problem 2

```
tol=1e-6;
max_its=100;
fcn='x^5-x^4+x-1';
trueroot=1;
dfcn='5*x^4-4*x^3+1';
inits=[-100; 0; 0.9;0.99;1.1;1.4;1000000];

disp(sprintf('\nNewtons estimate for root of %s with accuracy of
 %g',fcn,tol));
disp(sprintf('_initial_ \t _estimate_ \t _error_ \t _iterations_'))
for i=1:length(inits),

  [itiN(i),rootiN(i)]=newton(fcn,dfcn,inits(i),tol,max_its);
  disp(sprintf('%g\t\t %0.8f \t %0.5e \t %d',...
        inits(i),rootiN(i),abs(trueroot-rootiN(i)),itiN(i)));
end
% a. Newton's method is more efficient compared to Bisection
% method as the number of interations is less when the initialized
 point is
% closer to the true root. It seems that Newton's method took 2
 interations
% for 0 as an initialized point which is fast compared to 0.99, which
 is
% closer to the true root by 0.01, though it took 3 iterations.

% b. As the rate of convergence for Newton's method is quadratic, so
 the
% bumber of accurate digits roughly doubles at every step.


% c. Biscetion's itertations >= log((-1000000-1000000)/1.3x10^-12)/
log(2),
% which is approximately 60 itertations, which is better than Netwon's
 method as it
% took 67 iterations.


Newtons estimate for root of x^5-x^4+x-1 with accuracy of 1e-06
_initial_   _estimate_   _error_   _iterations_
-100   1.00000000   0.00000e+00   28
0   1.00000000   0.00000e+00   2
0.9   1.00000000   0.00000e+00   5
0.99   1.00000000   1.33227e-14   3
1.1   1.00000000   1.17906e-12   4
1.4   1.00000000   2.22045e-16   6
1e+06   1.00000000   1.27010e-13   67
```

Problem 3

```
tol=1e-6;
max_its=100;
fcn='x^5-x^4+x-1';
trueroot=1;
intervals=[0, 3; 0.5, 2.0; 0.9, 1.2];

disp(sprintf('\nSecant estimate for root of %s with accuracy of
 %g:',fcn,tol));
disp(sprintf('_interval_ \t _estimate_ \t _error_ \t _iterations_'))

for i=1:size(intervals,1),

 [itiS(i),rootiS(i)]=secant(fcn,intervals(i,1),intervals(i,2),tol,max_its);
  disp(sprintf('[%g,%g]    \t %0.8f \t %0.5e \t %d',...
        intervals(i,:),rootiS(i),abs(trueroot-rootiS(i)),itiS(i)));
end
% a. Secant method is more efficient than Bisection in every way,
 however,
% it is efficient in small intervals but not for large ones compared
 to
% Newton's method.

% b. No, as the intervals from [0,3] and [0.9,1.2] have different
 domains
% but they still have the same number of iterations although [0.9,1.2]
 is
% closer to the true root
```

```
Secant estimate for root of x^5-x^4+x-1 with accuracy of 1e-06:
_interval_   _estimate_   _error_   _iterations_
[0,3]      1.00000000   2.29172e-12   6
[0.5,2]      1.00000000   2.54086e-12   10
[0.9,1.2]      1.00000000   2.37077e-12   6
```

Problem 4

```
tol=1e-6;
max_its=100;
fcn='x^5-x^4+x-1';
trueroot=1;
intervals=[0, 3; 0.5, 2.0; 0.9, 1.2];
options=optimset('MaxIter',max_its,'TolX',tol);

disp(sprintf('\nBisection estimate for root of %s with accuracy of
 %g:',fcn,tol));
disp(sprintf('_interval_ \t _estimate_ \t _error_ \t _iterations_'))

for i=1:size(intervals,1),
[rootiF(i),fval,flag,output]=fzero(fcn,intervals(i,:),options);
```

```
itiF(i)=output.iterations;  disp(sprintf('[%g,%g]    \t %0.8f \t %0.5e
 \t %d',...
        intervals(i,:),rootiF(i),abs(trueroot-rootiF(i)),itiF(i)));
end
% Less iterations
tol=1e-6;
max_its=100;
fcn='x^5-x^4+x-1';
trueroot=1;
dfcn='5*x^4-4*x^3+1';
inits=[-100; 0; 0.9;0.99;1.1;1.4;1000000];
options=optimset('MaxIter',max_its,'TolX',tol);
disp(sprintf('\nNewtons estimate for root of %s with accuracy of
 %g',fcn,tol));
disp(sprintf('_initial_ \t _estimate_ \t _error_ \t _iterations_'))
for i=1:length(inits),

  [rootiF(i),fval,flag,output]=fzero(fcn,inits(i,:),options);
itiF(i)=output.iterations;
  disp(sprintf('%g\t\t %0.8f \t %0.5e \t %d',...
        inits(i),rootiF(i),abs(trueroot-rootiF(i)),itiF(i)));
end
% More error, Less iterations
tol=1e-6;
max_its=100;
fcn='x^5-x^4+x-1';
trueroot=1;
intervals=[0, 3; 0.5, 2.0; 0.9, 1.2];
options=optimset('MaxIter',max_its,'TolX',tol);
disp(sprintf('\nSecant estimate for root of %s with accuracy of
 %g:',fcn,tol));
disp(sprintf('_interval_ \t _estimate_ \t _error_ \t _iterations_'))

for i=1:size(intervals,1),
  [rootiF(i),fval,flag,output]=fzero(fcn,intervals(i,:),options);
itiF(i)=output.iterations;
  disp(sprintf('[%g,%g]    \t %0.8f \t %0.5e \t %d',...
        intervals(i,:),rootiF(i),abs(trueroot-rootiF(i)),itiF(i)));
end
% More error

% fzero's results are similar to Secant and it do as well as Biscetion
 as
% the error for both are 10^-07
```

```
Bisection estimate for root of x^5-x^4+x-1 with accuracy of 1e-06:
_interval_   _estimate_   _error_   _iterations_
[0,3]      0.99999941   5.91880e-07   7
[0.5,2]      1.00000097   9.71839e-07   8
[0.9,1.2]      1.00000040   4.04576e-07   5

Newtons estimate for root of x^5-x^4+x-1 with accuracy of 1e-06
_initial_   _estimate_   _error_   _iterations_
```

```
-100   1.00000000   5.55312e-10   22
0   1.00000000   0.00000e+00   1
0.9   1.00000000   1.96987e-11   3
0.99   1.00000001   6.06800e-09   4
1.1   1.00000000   9.20890e-10   5
1.4   1.00000019   1.87346e-07   5
1e+06   0.99999909   9.06677e-07   57

Secant estimate for root of x^5-x^4+x-1 with accuracy of 1e-06:
_interval_   _estimate_   _error_   _iterations_
[0,3]      0.99999941   5.91880e-07   7
[0.5,2]      1.00000097   9.71839e-07   8
[0.9,1.2]      1.00000040   4.04576e-07   5
```

Problem 5

```
disp(sprintf('\n'));
disp(roots(poly(1:21)));
% Poly's function returns a polynomial that has coeffieients of the
 input,
% so the polynomial for poly(1:21) would be x!/(x-22)!.

% Roots' function return the roots of the polynomial in general, so
% the roots for roots would be correct for the first 6 roots, then,
 however,
% roots will be less accurate as the degree of the polynomial
 increases due
% to the derivate is getting further from the main function.
```

```
   21.0007
   19.9942
   19.0196
   17.9613
   17.0416
   15.9823
   14.9872
   14.0246
   12.9849
   12.0020
   11.0036
    9.9972
    9.0011
    7.9998
    7.0000
    6.0000
    5.0000
    4.0000
    3.0000
    2.0000
    1.0000
```

Published with MATLAB® R2017b