Table of Contents


```
clear
clc
```

Problem 1

```
tol=1e-6;
max_its=100;
fcn='x^5-3*x^4+4*x^3-4*x^2+3*x -1';
fcn='-1 + x*(3+x*(-4+x*(4+x*(-3+x))))';
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. because the number of iteration depends on the interval and the
 error
% accuarcy

% b. Because this function is an ill-conditioned (unstable rootfinding
% problem) which leads to large errors in results for small errors in
% inputs.

% c. No, as the function is an ill-conditioned (unstable rootfinding
% porblem) and I have tested it by rewriting the function again in
% reveresed order, and nested form, and factoring it
% tol=1e-6;
% max_its=100;
% fcn='-1 + x*(3+x*(-4+x*(4+x*(-3+x))))';
% 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
```

```
Bisection estimate for root of -1 + x*(3+x*(-4+x*(4+x*(-3+x)))) with
 accuracy of 1e-06:
_interval_   _estimate_   _error_   _iterations_
[0,3]      1.00000548   5.48363e-06   21
[0.5,2]      1.00000453   4.52995e-06   20
[0.9,1.2]      0.99999790   2.09808e-06   18
```

Problem 2

```
tol=1e-6;
max_its=100;
fcn='x^5-3*x^4+4*x^3-4*x^2+3*x -1';
trueroot=1;
dfcn='5*x^4-12*x^3+12*x^2-8*x+3';
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 less efficient compared to Bisection's method
 when
% the initial iterate is close to the true root as the Bisection's
 method's
% iterations depend on the interval only compared to Newton's method's
% iterations which depends on the function and its derivative.

% b. The function is in ill-condition (unstable function), so we need
 to
% rewrite the new function would be g(x) = x - m f(x)/f'(x), as m is
 known
% which is the multiplicity of the root, which is 3, so the g(x) = x -
% ((x^5-3*x^4+4*x^3-4*x^2+3*x -1)/(5*x^4-12*x^3+12*x^2-8*x+3)).

% tol=1e-6;
% max_its=100;
% fcn='x -((x^5-3*x^4+4*x^3-4*x^2+3*x -1)/(5*x^4-12*x^3+12*x^2-8*x
+3))';
% trueroot=1;
% dfcn='(4*(x^2 + 1)*(5*x^2 -4*x +2))/(5*x^2 -2*x +3)^2';
```

```
% 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

% b. (con.) I didn't acheive the quadradic convergence.
```

```
Newtons estimate for root of x^5-3*x^4+4*x^3-4*x^2+3*x -1 with
 accuracy of 1e-06
_initial_   _estimate_   _error_   _iterations_
-100   1.00000110   1.09607e-06   53
0   0.99999628   3.72051e-06   31
0.9   0.99999630   3.70103e-06   31
0.99   0.99999731   2.69277e-06   24
1.1   1.00000677   6.77341e-06   25
1.4   1.00000821   8.21270e-06   30
1e+06   0.99999920   8.03039e-07   95
```

Problem 3

```
tol=1e-6;
max_its=100;
fcn='x^5-3*x^4+4*x^3-4*x^2+3*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's method is less efficient compared to Newton's and
 Bisection's
% methodas it takes as it takes more iterations than both of them with
% similar range of error.

% b. No
```

```
Secant estimate for root of x^5-3*x^4+4*x^3-4*x^2+3*x -1 with accuracy
 of 1e-06:
```

```
_interval_   _estimate_   _error_   _iterations_
[0,3]      0.99999539   4.60698e-06   45
[0.5,2]      0.99999305   6.94623e-06   41
[0.9,1.2]      0.99999530   4.69714e-06   38
```

Published with MATLAB® R2017b