MATH 391 **Worksheet 07** **F2019**

Root Finding - Bisection, Newton and Secant Methods

01. Bracket all the roots of the equations
a. $x - 3^{-x} = 0$ **b.** $x^3 - 2x^2 - 4x + 2 = 0$ **c.** $x\cos(x) - 2x^2 + 3x - 1 = 0$
One way to do it is to use MATLAB to plot the graph of the function, locate its x-intercept(s), then find intervals that contain each of those intercepts.

02. Consider the equation $e^x - 4x = 0, \quad 0 \le x \le 2$. Using the Bisection method, how many iterations are needed until the midpoint c_n, is an approximation of the root r, correct to 4 decimal places, i.e., $|r - c_n| < 0.5\,10^{-4}$.

03. Use the MATLAB function mybisection that appear at the end of this worksheet or your own function to find the unique root of $x^3 = 27$, accurate to within 10^{-5}, i.e., $|r - c_n| < 0.5\,10^{-5}$.

04. Repeat the previous problem with the equation $x - 3^{-x} = 0$

05. Use Newton's Method to approximate the only real root of the equation $5x^7 + 2x - 1 = 0$. Compute 10 iterates and numerically verify the quadratic convergence of the method. You may use the MATLAB function mynewton() that appear at the end of this worksheet to do the calculations.

06. Let $f(x) = (x+3)(x-1)^2$. Apply Newton's method to the equation $f(x) = 0$ with initial guesses
a. $x_1 = -4$ **b.** $x_1 = -2$ **c.** $x_1 = 0$
For each initial guess determine which of the two roots the method converges to. Comment on the rate of convergence.
What happens when you use $x_1 = -1$ as initial guess?

07. Write down Newton's algorithm for the equation $(x - r)^p = 0, \quad p > 1$. Show that for any initial guess x_1, Newton's sequence converges to the root r. What is the rate of convergence?

08. Write down the Secant algorithm for the equation $x^3 - a = 0, \quad a \in \mathbb{R}$. Simplify your formula completely.

09. Use the Secant method to determine the root r of the equation $2x = e^{-x}$ to 8 correct decimal places, i.e., $|r - x_n| < 0.5\,10^{-8}$. How many iterations did it take? You may use the MATLAB function mysecant() that appear at the end of this worksheet to do the calculations.

10. Solve for the 3^{rd} smallest positive root of the equation $x - \tan(x) = 0$, using the Bisection method, the Secant method and Newton's method. Make a table with successive iterates of each method in separate columns. Comment on the speed of the three methods. You may start by ploting $y = x - \tan(x)$ to determine a good bracketing interval of the root. You may use the MATLAB functions mybisection(), mynewton() and mysecant() that appear at the end of this worksheet to do your calculations.

11. What happens if Newton's method or the Secant method is applied to a linear equation $m\,x + b = 0,\ m \neq 0$?

12. Let $f(x) = \begin{cases} -\sqrt{-x} & \text{if} \quad x < 0 \\ \sqrt{x} & \text{if} \quad x \geq 0 \end{cases}$ and consider the equation $f(x) = 0$.

a. Write down Newton's algorithm and compute by hand the sequence generated by the algorithm. Does it converge?

b. Write down the Secant algorithm and compute by hand the sequence generated by the algorithm when $x_2 = -x_1$. Does it converge?

13. Repeat problem (12) in the following cases

a. $f(x) = x^{1/3}$

b. $f(x) = \begin{cases} -\,x^{2/3} & \text{if} \quad x < 0 \\ x^{2/3} & \text{if} \quad x \geq 0 \end{cases}$

14. A sequence $\left\{x_n\right\}_{n\geq 1}$ is said to converge superlinearly to r, if

$$\left|r - x_{n+1}\right| \leq \alpha_n \,\left|r - x_n\right|, \quad \forall n \in \mathbb{N}, \qquad \text{where} \qquad \lim_{n\to+\infty} \alpha_n = 0$$

Show that if $\left\{x_n\right\}_{n\geq 1}$ converges superlinearly to r, then $\displaystyle\lim_{n\to+\infty} \frac{|r - x_n|}{\left|x_{n+1} - x_n\right|} = 1$

```
function [X,r,n] = mybisection(a,b,fname,tol)
%
% Input
% a & b endpoints of the interval that brackets the root of
%      f(x)=0. They must satisfy f(a)*f(b)<0
% fname handle of the function f(x) whose root is being computed.
% tol  tolerance used to terminate the iterations. If not
%    provided, it will be set to 0.
% Output
% X   last interval that brackets the root.
% r   an approximate value of the root.
% n   the number of iterations performed
%
% Call [X r n]=bisection(2.0, 5.0,@myf, 10^(-8));
%       here myf is the MATLAB name of the function f(x)
%
% define the left and right endpoints of the bracketing interval
 XL=a; XR=b;
% Initialize the number of iterations
 n=0;
% Check if the interval [a,b] brackets the root
 fL=fname(XL); fR=fname(XR);
 if fL*fR > 0
    disp('original interval does not bracket the root')
    X=[XL XR]; r=(XL+XR)/2;
    return
 end
%
% define the stopping tolerance
 if nargin==3
    tol=0;
 end
 delta=max(tol,eps*max(abs(a),abs(b)));
%
% Bissect until an interval that brackets the root with length
% smaller than delta is found.
 while abs(XR-XL)>delta
       XM=(XL+XR)/2; fM=fname(XM); n=n+1;
       if fL*fM <= 0
          XR=XM; fR=fM;
       else
          XL=XM; fL=fM;
       end
 end
%
% take the midpoint of [XL,XR] as an approximate value of the root
 r=(XL+XR)/2; X=[XL XR];
```

```
function [X, neval, status] = mynewton(xin,fname,tol,nmax)
%
% finds the root r of f(x)=0 by using Newton's algorithm
% x_1=xin,  x_n=x_{n-1}-f(x_{n-1})/f'(x_{n-1}), for n=2,3,4,...
%
% Input
% xin    the initial guess at the root r
% fname  handle of the function f(x) whose zero is being approximated
%        It should evaluate both f(x) and f'(x) at any given x.
% tol    tolerance used to terminate the iterations whenever
%        the increment |f(x)/f'(x)| < tol . If not provided
%        it will be set to 10^(-5).
% nmax   maximum number of iterations that can be performed.
%        If not provided, it will be set to 30.
% Output
% X      the sequence of approximations x_1, x_2, ... , x_n
%        of the root that have been generated.
% R      the relative error R(i)=abs( (X(i)-X(i-1))/X(i) )
%        it could be used as a stopping criteria.
% neval  total number of functions evaluations of both f(x)
%        and f'(x)
% status has value 1 if the iterations were successful and -1
%        otherwise.
%
% Call [X n status] = mynewton(1.2,@myf, 10^(-8), 120);
%
% evaluate f and f' at x=xin and compute the ratio f/f'
 x = xin; y = fname(x); f = y(1); fp = y(2); ratio = f/fp;
% initialize N, X and the relative error R
 N = 1; X = [x]; R = 1.0;
%
 p=1;
 if p==1
    disp(' x                    rel. error   f(x)         fp(x) ')
    disp('=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=')
    disp(sprintf('%18.16f    %4.3e    %4.3e   %4.3e',x,R,abs(f),abs(fp)))
 end
%
% define the tolerance and the maximum number of iterations
  if nargin==2
     tol = 10^(-5); nmax = 30;
  elseif nargin==3
     nmax = 30;
  end
%
 while (abs(ratio) > tol) & (N<nmax)
  R=x;
  x = x - ratio;
  X = [X; x];
  % computes the relative error from two successive iterates.
  if x~=0
     R=abs( (x-R)/x );
  else
     R=0;
  end
  y = fname(x); f=y(1); fp=y(2);
  N=N+1;
  ratio = f/fp;
  if p==1
     disp(sprintf('%18.16f    %4.3e    %4.3e   %4.3e',x,R,abs(f),abs(fp)))
  end
 end
%
 status = 1;
 if N >= nmax
    status = -1;
    disp(' ==- max. number iterations reached and no conv. -== ')
 end
 neval = 2*N;
```

```
 function [X,r,neval] = mysecant(a,b,fname,tol,nmax)
%
% Finds the root r of the equation f(x)=0, using the secant method.
% Input
% a & b endpoints of the interval that brackets the root r of the
%       equation f(x)=0.
% fname handle of the function f(x) whose zero is being computed
% tol   tolerance used to terminate the iterations whenever the
%       distance between two successive iterates is less than tol
% Output
% X     the successive iterates generated by the method
% r     approximate value of the root.
% neval number of function evaluations.
%
% Call  [X,r,neval] = mysecant(1.5,3.0,@myf,10^(-5),50)
%       Here myf is the MATLAB name of the function f(x)
%
% Check if the interval [a,b] brackets the root
  xa = a; fxa = fname(xa); xb = b; fxb = fname(xb);
  if fxa*fxb > 0
     disp('original interval does not bracket the root')
     return
  end
% Define the tolerance and the maximum number of iterations
  if nargin == 3
     tol = 0; nmax = 50;
  elseif nargin == 4
     nmax = 50;
  end
  delta=tol+eps*max(abs(xa),abs(xb));
% initialize the number of evaluatio, X and the relative error R
  N = 2; X = [xa; xb]; R = 1;
% set p=0 if print out of X, R and f is not wanted
  p = 1;
  if p==1
    disp(' x                    rel. error   f(x) ')
    disp('=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=~=')
    disp(sprintf('%18.16f    %4.3e    %4.3e',xa,R,abs(fxa)))
    disp(sprintf('%18.16f    %4.3e    %4.3e',xb,R,abs(fxb)))
 end
%
% define the distance between two successive iterates
  deltax=abs(xb-xa);
  %%%
  while (deltax>delta) & (N<nmax)
         % find the x-intersect of the secant line
     ratio = fxa/((fxb-fxa)/(xb-xa)); xi = xa-ratio;
   % reset xa, xb, fxa, fxb
     xa=xb; fxa=fxb; xb=xi; fxb=fname(xb);
   % update neval, X and deltax
     N = N+1; X=[X; xi]; deltax=abs(xb-xa);
   % compute the relative error
     if xb ~= 0
        R = abs( (xb-xa)/xb );
     else
        R = 1;
     end
     if p==1
     disp(sprintf('%18.16f    %4.3e    %4.3e',xb,R,abs(fxb)))
     end
  end
  %%%
% determine whether there was convergence or not
  if deltax <= delta
     status = 1;
  else
     status = -1;
     disp(' ==- max. number iterations reached and no conv. -== ')
  end
% take the last iterate as an approximation of the root
  r = xb;
% define the total number of iterations
  neval = N;
```