Your Signature ______________________

Instructions:
Please write your name on every page.
You may use the back side of every page for rough work or writing answers.
Maximum time is 2 hours and 30 min.
Show all your work. Correct answers with insufficient or incorrect work will not get any credit.

Score

1.	(20)	
2.	(20)	
3.	(20)	
4.	(20)	
5.	(20)	
Total.	(100)	

Extra sheets attached(if any): ______________________

1.(a) Suppose $A = (a_{i,j})_{n\times n}$ is a tri-diagonal matrix (i.e. $a_{i,j} = 0$ whenever $|i-j| \geq 2$). Assuming you never need to swap rows, it is known that Gaussian elimination to solve $Ax = b$ for a given column vector $b_{n\times 1}$, requires $O(n^k)$ flops. Find k.

(b)Suppose you want to compute

$$x_1 = 1 - \sqrt{1 + \frac{1}{10^{20}}}, \qquad x_2 = 1 + \sqrt{1 + \frac{1}{10^{20}}}.$$

There is a loss of significance in one of the above compuation. Which one of x_1 and x_2 will suffer from this loss of significance ? Suggest a method to avoid this difficulty.

2 Below are two function files. Please fill in the blanks so as to ensure that the programs run correctly in OCTAVE.

(a)

```
function ssum = sinser(x,tol,n)
% sinser  Evaluate the series representation of the sine function
%
% Synopsis: ssum = sinser(x)
%           ssum = sinser(x,tol)
%           ssum = sinser(x,tol,n)
%
% Input:    x   = argument of the sine function, i.e., compute sin(x)
%           tol = (optional) tolerance on accumulated sum. Default:  tol = 5e-9
%                 Series is terminated when abs(T_k/S_k) < delta.   T_k is the
%                 kth term and S_k is the sum after the kth term is added.
%           n   = (optional) maximum number of terms. Default: n = 15
%
% Output:   ssum = value of series sum after nterms or tolerance is met

if nargin <   ______,  tol = 5e-9;  end
if nargin <   ______,  n = 15;      end

term =   ______;  ssum =   ______;         %  Initialize series
fprintf('Series approximation to sin(%f)\n\n  k      term         ssum\n',x);
fprintf('%3d  %11.3e  %12.8f\n',1,term,ssum);

for k=3:2:(2*n-1)
  term = -term * x*x/(k*(k-1));                  %  Next term in the series
  ssum =  ______;
  fprintf('%3d  %11.3e  %12.8f\n',k,term,ssum);
  if abs( ______)<tol, ______;  end             %  True at convergence
end
fprintf('\nTruncation error after %d terms is %g\n\n',(k+1)/2,abs( ______));
```

(b)

```
{ function r = ______________________(fun,x0,xtol,ftol,verbose)
% newton     Newton's method to find a root of the scalar equation f(x) = 0
%
% Synopsis:  r = newton(fun,x0)
%            r = newton(fun,x0,xtol)
%            r = newton(fun,x0,xtol,ftol)
%            r = newton(fun,x0,xtol,ftol,verbose)
%
% Input:  fun     = (string) name of mfile that returns f(x) and f'(x).
%         x0      = initial guess
%         xtol    = (optional) absolute tolerance on x.    Default: xtol=5*eps
%         ftol    = (optional) absolute tolerance on f(x). Default: ftol=5*eps
%         verbose = (optional) flag.  Default: verbose=0, no printing.
%
%  Output:  r = the root of the function

if nargin < ___,  xtol = 5*eps;  end
if nargin < ___,  ftol = 5*eps;  end
if nargin < ___,  verbose = 0;   end
xeps = max(xtol,5*eps);  feps = max(ftol,5*eps);  %  Smallest tols are 5*eps

if verbose
  fprintf('\nNewton iterations for %s.m\n',fun);
  fprintf('  k      f(x)         dfdx         x(k+1)\n');
end

x = ____;  k = ____;  maxit = 15;    %  Initial guess, current and max iterations
while k <= maxit
  k = k + 1;
  [f,dfdx] = feval(fun,x);      %   Returns f( x(k-1) ) and f'( x(k-1) )

  dx = ______;

  x = _______;

  if verbose, fprintf('%3d %12.3e %12.3e %18.14f\n',k,f,dfdx,x);  end

  if ( abs(f) < _____ ) | ( abs(dx) < ______ ),  r = _____; ________;  end
end
warning(sprintf('root not found within tolerance after %d iterations\n',k));
}
```

3. Find the $PA = LU$ factorization (i.e perform LU factorisation with partial pivot) of the matrix $A = \begin{bmatrix} \frac{1}{2} & 1 & -1 \\ 1 & -1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$.

4. Consider the following:

Algorithm:

```
Input f
Initialise n ,a, b, such that f(a) and f(b) have
opposite signs and a < b
x(0)= a
x(1)= b
k =1
  while k <= n
    x(k+1) = x(k-1) - f(x(k-1)) (x(k) - x(k-1))/(f(x(k))-f(x(k-1)))
      if sign(f(x(k+1))) = sign(f(a))
        a = x(k+1)
      elseif sign(f(x(k+1))) = sign(f(b))
         b = x(k+1)
      end    k = k +1
end
```

(a) Describe what the algorithm is doing in words with the help of a picture.

(b) Construct an example of f (*picture is enough*) such that $f(a) < 0$ and $f(b) > 0$. The above algorithm keeps b fixed and moves the left end point of interval closer to the desired conclusion.

5. Prove the following theorem: Suppose $g : [a, b] \to [a, b]$ and $g \in C^1[a, b]$ with $| g'(x) | < 1$ for $x \in [a, b]$. Let $x_0 \in [a, b]$ and $x_k = g(x_{k-1}), k \geq 1$. Show that $\exists \xi \in [a, b]$ such that

$$\lim_{k \to \infty} x_k = \xi, \quad \text{and} \quad g(\xi) = \xi.$$