Problem 2

```
% a
% Test my function
% You can find the function at the end
A = [1,2,3;40,35,6;7,8,9];
b = [1;2;2];
x = gaussian_partial_pivoting(A,b)
```

```
x = 3×1
   -0.6458
    0.7917
    0.0208
```

```
linsolve(A,b)
```

```
ans = 3×1
   -0.6458
    0.7917
    0.0208
```

```
% b
% Check my answer in the paper
A = [3,7;6,1]
```

```
A = 2×2
     3     7
     6     1
```

```
b = [1;-11]
```

```
b = 2×1
     1
   -11
```

```
linsolve(A,b)
```

```
ans = 2×1
    -2
     1
```

```
gaussian_partial_pivoting(A,b)
```

```
ans = 2×1
    -2
     1
```

Problem 3

```
% b
A = [5,1,-1;
    3,6,2;
    2,-6,9]
```

```
A = 3×3
     5     1    -1
     3     6     2
     2    -6     9
```

```
b = [1;1;1]
```

```
b = 3×1
     1
     1
     1
```

```
n = length(b);
x0 = zeros(n,1);

% Check the convergence of jacobi and gaussian seidel
% we see that both method converge to the solution solved by matlab
k =1000000; % number of iteration

%a
jacobi(A,b, k, x0) % jacobi method
```

```
ans = 3×1
    0.2107
    0.0326
    0.0861
```

```
%b
gauss_seidel(A,b,k,x0) % gaussian seidel method
```

```
ans = 3×1
    0.2107
    0.0326
    0.0861
```

```
linsolve(A,b) % using matlab
```

```
ans = 3×1
    0.2107
    0.0326
    0.0861
```

Problem 5

```
% a two-point forward-difference
% f'(x) = 1/x
x = 1;
f = @(x) log(x);
```

```
h = 0.1;
diff_f_1 = (f(x+h) - f(x))/h % 2-point forward-difference
```

```
diff_f_1 = 0.9531
```

```
% Approximation error
error = abs(diff_f_1 - 1/x)
```

```
error = 0.0469
```

```
% three-point centered-difference formula
% f' = exp(x)
x = 0;
f = @(x) exp(x);
h = 0.1;
diff_f_1 = (f(x+h) - f(x-h))/(2*h) % 3-point centered-difference
```

```
diff_f_1 = 1.0017
```

```


% Aprroximation error
error = abs(diff_f_1 - exp(x))
```

```
error = 0.0017
```

```
function x=gaussian_partial_pivoting(A, b)
% This function apply gaussian elimination with partial pivoting
a = A;
n = length(b);
for j = 1 : n -1
    if abs ( a (j ,j )) < eps
        error ( "zero pivot encountered");
    end

    % Find the largest pivot
    r = j;
    for i = j+1:n
        if abs(a(i,j))>abs(a(r,j))
            r = i;
        end
    end

    % Exchange row in A
    temp_a = a(j, :);
    a(j,:) = a(r,:);
    a(r,:) = temp_a;

    % Exchange row in b
    temp_b = b(j);
    b(j) = b(r);
```

```
    b(r) = temp_b;

    % For ward elimination
    for i = j + 1 : n
        mult = a (i , j ) / a (j , j );
        a(i,:) = a(i,:) - mult * a (j,:);
        b ( i ) = b( i ) - mult * b ( j );

    end
end

% Backward substitution
x = zeros(n,1);
for i = n : -1 : 1
    for j = i +1 : n
        b ( i ) = b( i ) - a (i , j ) * x ( j );
    end
        x ( i ) = b( i ) / a (i , i );
end

end

% Gauss-Seidel method
% Estimates solution for equation Ax=b
% Input: coefficient matrix a,
%        right-hand-side vector b,
%        number of interations k,
%        initial guess x0
% Output: solution x.
function x = gauss_seidel(a,b,k,x0)
n=length(b);
d=diag(a);
l = tril(a, -1);
u = triu(a, 1);
x = x0.*ones(n,1);
xi = x;
for j=1:k
    for i=1:n
        xi(i) = (b(i) - u(i,:)*x - l(i,:)*xi) /d(i);
    end
    x = xi;
end
end

% Jacobi Method
% Estimates solution for equation Ax=b, uses zero vector as an initial
% guess
% Inputs: full or sparse matrix a,
%         right-hand-side vector b,
%         number of Jacobi iterations k,
```

```
%          initial guess x0
% Output: solution x
function x = jacobi(a,b,k,x0)
n=length(b);      % find n
d=diag(diag(a));        % extract diagonal of a
r=a-d;      % r is the remainder
x=x0.*ones(n,1);     % initialize vector x
for j=1:k         % loop for Jacobi iteration
  x=d\(b-r*x);
end               % End of Jacobi iteration loop
end
```