Numerical Technique(MA202)

Lab Assignment 6

March 1, 2020

Course Instructor: Dr. Jignesh S. Bhatt

Student and ID : Pushkar Patel(201851094)

---

1 Solve the following linear system

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \\ 3 & 4 & -2 \end{bmatrix} b = \begin{bmatrix} 4 \\ 7 \\ 9 \end{bmatrix}$$

1.1 Using Gauss elimination

1.2 Using LU decomposition

1.3 Using Gauss elimination+partial pivoting

2 Solve the following linear system

$$A = \begin{bmatrix} 1 & 2 & 2 & 1 \\ 2 & 2 & 4 & 2 \\ 1 & 3 & 2 & 5 \\ 2 & 6 & 5 & 8 \end{bmatrix} b = \begin{bmatrix} 1 \\ 0 \\ 2 \\ 4 \end{bmatrix}$$

2.1 Using Jacobi method of iteration

2.2 Using Gauss-Siedel method

Matlab script to calculate solution of Linear Equations

Table of Contents


```
clc;
clear;
close all;

A = [1 1 1; 2 1 3; 3 4 -2];
b =[4;7;9];
```

Using Gauss Elemination

```
x = solutionofLinearEquations(A,b);
disp("Solution using Gauss Elemination:");
disp(x);
```

```
Solution using Gauss Elemination:
     1
     2
     1
```

Using LU Decomposition

```
x = solutionofLinearEquations(A,b,2);
disp("Solution using LU Decomposition:");
disp(x);
```

```
Solution using LU Decomposition:
     1
     2
     1
```

Using Gauss elimination + partial pivoting

```
x = solutionofLinearEquations(A,b,3);
disp("Solution using Gauss elimination + partial pivoting:");
disp(x);
```

```
Solution using Gauss elimination + partial pivoting:
    1.0000
    2.0000
    1.0000
```

Published with MATLAB® R2019b

Solution of linear equations

Table of Contents

Function define

```
function fval = solutionofLinearEquations(a,b,choice)
    if ~exist('choice','var')
     % third parameter does not exist, so default it to something
      choice = 1;
       % By Default method is Gauss Elemination
    end
 switch choice
 case 1
 fval = GaussElemination(a, b);
 case 2
 fval = LUdecomposition(a,b);
 case 3
 fval = partialpivoting(a, b);
 end
end
```

```
Not enough input arguments.

Error in solutionofLinearEquations (line 11)
 fval = GaussElemination(a, b);
```

Gauss Elimination

```
function fval = GaussElemination(A,b)

%get augumented matrix
Ab =[A,b];
%Row Operation
% Rj =Rj-k(i,j)*Ri where k(i,j) = A(j,i)/A(i,i)
n = length(A);
%A(1,1) as pivot element
for i =2:n
    k = Ab(i,1)/Ab(1,1);
    Ab(i,:)= Ab(i,:)-k*Ab(1,:);
end
%A(2,2) as pivot element
i =n;
k = Ab(i,2)/Ab(2,2);
```

```
Ab(i,:)= Ab(i,:)-k*Ab(2,:);

%A(3,3 ) as pivot element

%Back-Subsituation
fval = zeros(n,1);
%x(3)=Ab(3,4)/Ab(3,3);
for i =n :-1:1
    %x(2)= (Ab(2,4)-Ab(2,3)*x(3))/Ab(2,2);
    fval(i)= (Ab(i,end)-Ab(i,i+1:n)*fval(i+1:n))/Ab(i,i);
    %x(1) = (Ab(1,4)-(Ab(1,3)*x(3)+Ab(1,2)*x(2)))/Ab(1,1);
    %x(1) = (Ab(1,4)-(Ab(1,1+1:n)*x(1+1:n))/Ab(1,1);
end
end
```

LU Decomposition

```
function fval = LUdecomposition(A,b)

%get augumented matrix
Ab =[A,b];
n = length(A);
L =eye(n);

%Row Operation
% Rj =Rj-k(i,j)*Ri where k(i,j) = A(j,i)/A(i,i)

%A(1,1) as pivot element
for i =2:n
    k = Ab(i,1)/Ab(1,1);
    L(i, 1)=k;
    Ab(i,:)= Ab(i,:)-k*Ab(1,:);

end

%A(2,2) as pivot element
i =n;
k = Ab(i,2)/Ab(2,2);
L(i,2)=k;
Ab(i,:)= Ab(i,:)-k*Ab(2,:);

%A(3,3 ) as pivot element

U = Ab(1:n,1:n);
y = inv(L)*b;
fval = inv(U)*y ;
end
```

Gauss elimination + partial pivoting

```
function fval = partialpivoting(A,b)
    %get augumented matrix
```

```
Ab =[A,b];
n = length(A);

%A(1,1) as pivot element
% Ensure A(1,1) is largest element in column-1
col1 = Ab(:,1);
[dummy,idx]= max(col1);
dummy =Ab(1,:);
Ab(1,:)=Ab(idx,:);
Ab(idx,:)= dummy;

for i =2:n
    k = Ab(i,1)/Ab(1,1);
    Ab(i,:)= Ab(i,:)-k*Ab(1,:);

end

%A(2,2) as pivot element
% Ensure A(2,2) is largest element in column-2
col2 = Ab(2:end,2);
[dummy,idx]= max(col2);
dummy =Ab(2,:);
Ab(2,:)=Ab(idx,:);
Ab(idx,:)= dummy;

i =3;
k = Ab(i,2)/Ab(2,2);
Ab(i,:)= Ab(i,:)-k*Ab(2,:);

%A(3,3 ) as pivot element

% Back-Subsituation

fval = zeros(n,1);

%x(3)=Ab(3,4)/Ab(3,3);
for i =n :-1:1
    fval(i)= (Ab(i,end)-Ab(i,i+1:n)*fval(i+1:n))/Ab(i,i);
end

end
```

Published with MATLAB® R2019b

Matlab script to calculate solution of Linear Equations

Table of Contents


```
clc;
clear;
close all;

A = [1 2 2 1; 2 2 4 2;1 3 2 5;2 6 5 8];
b =[1;0;2;4];
```

Using Gauss Seidel Method

```
x = IterativemethodofLS(A,b,1,1e-3,100);
disp("Solution using Gauss Seidel Method :");
disp(x);
```

```
Solution using Gauss Seidel Method :
   1.0e+30 *

   -5.0703
   -1.2677
    7.6056
   -2.5352
```

Using Jacobi Method

```
x = IterativemethodofLS(A,b,2,1e-3,100);
disp("Solution using Jacobi Method :");
disp(x);
```

```
Solution using Jacobi Method :
   1.0e+54 *

   -1.1841
   -0.9701
   -0.8970
   -0.4485
```

Published with MATLAB® R2019b

Solution of linear equations using Iterative Method

Table of Contents

Function define

```
function fval= IterativemethodofLS(a,b,choice,tol,maxItr)
 switch choice
 case 1
 fval = gaussSeidel(a, b,tol,maxItr);
 case 2
 fval = Jacobi(a,b,tol,maxItr);
 end
end

Not enough input arguments.

Error in IterativemethodofLS (line 4)
 switch choice
```

Gauss Seidel Method

```
function sol= gaussSeidel(A,b,tol,maxitr)

   % Here co-efficient matrix A must be 'strictly diagonally dominant
 matrix'

   % tol is maximum bearable tolerance in answer

   % maxitr is limit of iterations

   n=length(A);

   Xnext=zeros(n,1);      % assuming initial approximation as zero
 vector

   for loop=1:maxitr

       Xcurr=Xnext;

       for i=1:n
           temp=0;
```

```
                for j=1:n

                   if(i~=j)
                   temp=temp+(A(i,j)*Xnext(j));
                   end

                end

                Xnext(i)=(b(i)-temp)/A(i,i);
            end


           error=Xnext - Xcurr;

           err=norm(error);

               if err<=tol
                   sol=Xnext;
                   break;
               end

       end
       sol=Xnext;
   end
```

Jacobi Method

```
function fval= Jacobi(A,b,tol,maxitr)

    % Here co-efficient matrix A must be 'strictly diagonally dominant
 matrix'

    %tol is maximum bearable tolerance in answer

    % maxitr is limit of iterations

    n=length(A);

    Xcurr =zeros(n,1);          % assuming initial approximation as
 zero vector
    Xnext=zeros(n,1);

    for loop=1:maxitr

         for i=1:n
          temp=0;
             for j=1:n                              %
                                                    %
                if(i~=j)                            %
                temp=temp+(A(i,j)*Xcurr(j));        %   This loop
 calculates #k#j a(k,j)*x(j)
                end                                 %
                                                    %
```

```
            end                                   %

            Xnext(i)=(b(i)-temp)/A(i,i);
        end

        error=Xnext-Xcurr;
        err=norm(error);

            if err<=tol
                fval=Xnext;
                break;
            end

        Xcurr=Xnext;

    end
    fval=Xnext;
end
```

Published with MATLAB® R2019b