



**MATLAB Code**

**Gauss Elimination Method**

B=zeros(4,1);

B(1)=14;

B(2)=28;

B(3)=17.5;

B(4)=10.5;

C=[0.7 0.15 0.03 0.02;

0.1 0.65 0.06 0.04;

0.14 0.15 0.77 0.09;

0.06 0.05 0.14 0.85];

A = [C B];

[r,c]=size(A);

% sin -> whether the matrix is singular(sin=1) or non-singular(sin=0)

singular=false;

for i=1:r

% finding the i-th pivot:

% partial pivoting:

if(i<r)% do partial pivoting only if there are any row below the current row

imax=i; %index of the element with maximum value

max=A(i,i); %value of that element

for k=i+1:r

% finding the max

if abs(A(k,i))>abs(max)

max=A(k,i);

imax=k;

end

end

%swap the rows

A([i,imax],:)=A([imax,i],:);

end

if A(i,i)==0

% matrix is singular

singular=true;

end

% do for all remaining elements in current row

for j=i+1:r

A(j,:)=A(j,:)-A(i,:)\*A(j,i)/A(i,i);

A(j,i)=0; % fill lower triangular matrix with zeros

end

end

% if matrix is non-singular

if singular==false

sol=zeros(r,1);% solution array

% backward susbstitution

for i=r:-1:1

s=A(i,c);% s-> it will become the value of x(i)

for j=r:-1:i+1

s=s-A(i,j)\*sol(j,1);% this value needs to be removed from s

end

sol(i,1)=s/A(i,i);% divide by coeff of x(i)

end

else

disp('Matrix is Singular.');

end

**Gauss-Seidel Method**

B=zeros(4,1);

B(1)=14;

B(2)=28;

B(3)=17.5;

B(4)=10.5;

A=[0.7 0.15 0.03 0.02;

0.1 0.65 0.06 0.04;

0.14 0.15 0.77 0.09;

0.06 0.05 0.14 0.85];

X=zeros(4,1);

tol=0.0001; % assumed tolerance since nothing is given

iterations=0;

while true

temp=X;

X(1)=(B(1)-A(1,2)\*X(2)-A(1,3)\*X(3)-A(1,4)\*X(4))/A(1,1);

X(2)=(B(2)-A(2,1)\*X(1)-A(2,3)\*X(3)-A(2,4)\*X(4))/A(2,2);

X(3)=(B(3)-A(3,1)\*X(1)-A(3,2)\*X(2)-A(3,4)\*X(4))/A(3,3);

X(4)=(B(4)-A(4,1)\*X(1)-A(4,2)\*X(2)-A(4,3)\*X(3))/A(4,4);

iterations=iterations+1;

fprintf('The iteration vector is: [');

fprintf('%g ',X);

fprintf(']\n');

ctr\_false=0;ctr\_true=0;

for i=1:4

if(abs(X(i)-temp(i))>tol)

ctr\_false=ctr\_false+1; % to count number of times tolerance is not met

else

ctr\_true=ctr\_true+1;

end

end

if(ctr\_false>0)

continue; % continuing iterations when tolerance is not met

else

break; % stopping iterations when tolerance met for all variables

end

end



