```
A = [
    2,10,8,8,6;
    1,4,-2,4,-1;
    0,2,3,2,1;
    3,8,3,10,9;
    1,4,1,2,1;
    ];
b=[52,14,12,51,15]';
A_o=A; %Since we overwrite A

n=size(A,1);

%Test 1
%{
[L,U,P] = lu(A)
x_m =A\b
P*A -L*U
%}

[A, intch,issingular] = Gauss(A);
A
```

```
A = 5×5
    3.0000    8.0000    3.0000   10.0000    9.0000
    0.6667    4.6667    6.0000    1.3333         0
    0.3333    0.2857   -4.7143    0.2857   -4.0000
    0.3333    0.2857    0.3636   -1.8182   -0.5455
         0    0.4286   -0.0909   -0.8000    0.2000
```

```
intch'
```

```
ans = 1×5
     4     4     4     5     5
```

```

%Test 2
%{
L_A = tril(A,  -1) + eye(n)
U_A = triu(A)
L_A * U_A
Pm=GetP(intch)
Pm*A_o-L*U %OK!
%}
[x, issingular] = solve(A,intch,b);
x'
```

```
ans = 1×5
    1.0000    2.0000    1.0000    2.0000    1.0000
```

```

%Test 3
%Pm*A_o*x - Pm*b %OK!
%x_1 = A_o \ b %OK!
```

```
%Now run on eye(n) to calculate A^{-1}
use_optimized =true;
A_inv = eye(n); %Solve in place col by col
for i=1:n
    b=A_inv(:,i);
    if use_optimized
        [x,issingular] = solveOpt(A,intch,b);
    else
        [x,issingular] = solve(A,intch,b);
    end
    A_inv(:,i) = x;
end
A_inv
```

```
A_inv = 5×5
   12.3333    3.3333  -21.6667   -3.6667  -16.0000
   -3.5000   -1.0000    6.0000    1.0000    5.0000
    2.6667    0.6667   -4.3333   -0.8333   -3.5000
    1.0000    0.5000   -1.5000   -0.2500   -1.7500
   -3.0000   -1.0000    5.0000    1.0000    4.0000
```

```
A_o*A_inv
```

```
ans = 5×5
    1.0000    0.0000    0.0000    0.0000    0.0000
   -0.0000    1.0000    0.0000    0.0000    0.0000
         0   -0.0000    1.0000         0    0.0000
   -0.0000         0   -0.0000    1.0000    0.0000
   -0.0000    0.0000    0.0000    0.0000    1.0000
```

```
function P = GetP(intch)
    n=size(intch,1)
    P=eye(size(intch,1))
    for i =1:n-1
        %Permute row i,intch(i)
        P([i intch(i)],:)=P([intch(i) i],:);
    end
end
```

```
%Expects a column vector
function [x,issingular] =solve(A,intch,b)
    issingular=false;
    n=size(b,1);
    assert(size(A,1)==n)
    for k=1:n-1
        m=intch(k);
        b_k=b(k);
        b(k)=b(m);
        b(m)=b_k;
    end
    for j=1:n-1 %Forward subst loop
        for i=j+1:n
            b(i)=b(i)-A(i,j)*b(j);
        end
    end
    for j=n:-1:1
        if A(j,j)==0
            issingular=true;
            x=nan;
            return
        end
        b(j)=b(j)/A(j,j);
        for i=1:j-1
            b(i)=b(i)-A(i,j)*b(j);
        end
    end
    x=b;
end

%Expects a column vector- takes into account leading zeros in forward subt
function [x,issingular] =solveOpt(A,intch,b)
    issingular=false;
    n=size(b,1);
    assert(size(A,1)==n)
    for k=1:n-1
        m=intch(k);
        b_k=b(k);
        b(k)=b(m);
        b(m)=b_k;
    end
    first_non_zero_idx = find(b~=0,1,'first');
    for j=first_non_zero_idx:n-1 %Forward subst loop
        for i=j+1:n
            b(i)=b(i)-A(i,j)*b(j);
        end
    end
    for j=n:-1:1
        if A(j,j)==0
            issingular=true;
            x=nan;
            return
        end
        b(j)=b(j)/A(j,j);
        for i=1:j-1
```

```
            b(i)=b(i)-A(i,j)*b(j);
        end
    end
    x=b;
end


function [A, intch,issingular] = Gauss(A)
    n = size(A,1);
    assert(n==size(A,2)); % Make sure we're operating on nxn
    issingular = false;
    intch= zeros(n,1);
    for k = 1:n-1
        amax = max(abs(A(k:n,k)));
        if amax ==0
            issingular=true;
            intch(k)=0;
        else
            %Find the location of biggest pivot in first col of A_kk
            m=k+find(abs(A(k:n,k))==amax,1,'first')-1;
            intch(k)=m;
            if m ~=k %Then permute rows
                A_m=A(m,1:n);
                A(m,1:n)=A(k,1:n);
                A(k,1:n)=A_m;
            end
            for i=k+1:n %Calculate multipliers
                A(i,k)=A(i,k)/A(k,k);
            end
            A_k=A(k,k+1:n);
            for i=k+1:n %Do row subtractions
                mult=A(i,k);
                A_i=A(i,k+1:n);
                A(i,k+1:n)=A_i-mult*A_k;
            end
        end
    end
    if A(n,n)==0
        issingular=true;
        intch(n)=0;
    else
        intch(n)=n;
    end
end
```