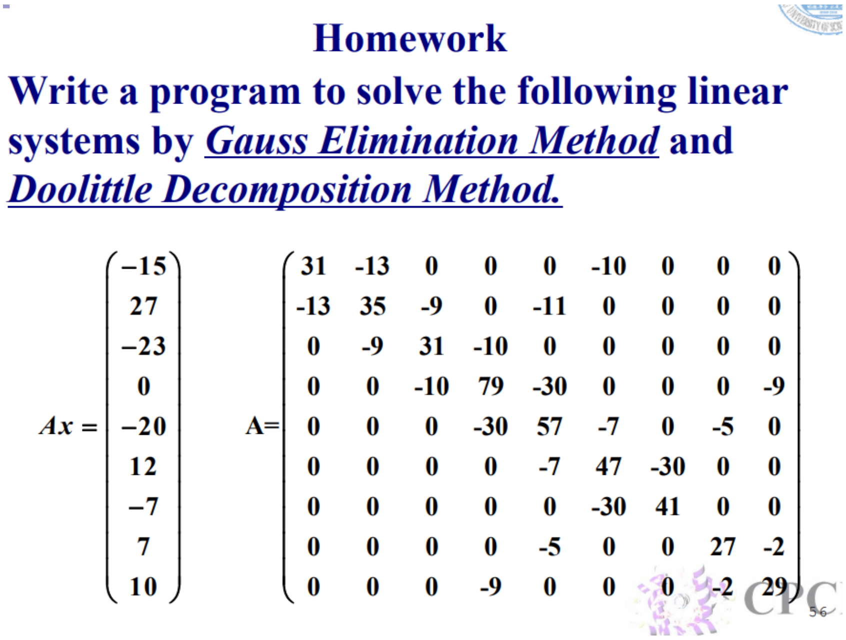
The third schoolwork of Computational Physics

万炫均 物理 1701 U201710170

Description of this chapter:

For this chapter, we try various computing methods to find the solutions of a series of linear equations. We usually use Matrices to represent the equations and transform them to get the solutions. Both transformative and iterative methods are used.

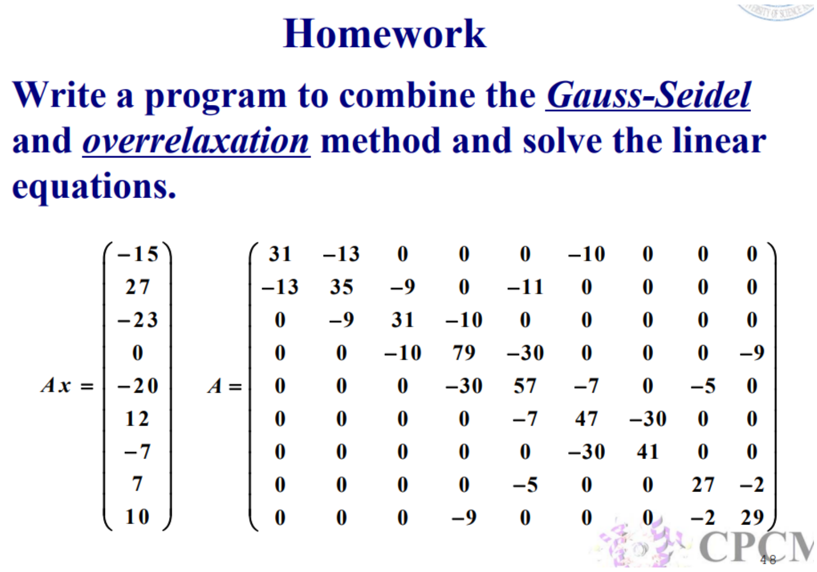
- **Description of the problem**



Homework

Write a program to solve the following linear systems by *Gauss Elimination Method* and *Doolittle Decomposition Method.*

$$Ax=\begin{pmatrix}-15\\27\\-23\\0\\-20\\12\\-7\\7\\10\end{pmatrix}\qquad A=\begin{pmatrix}31&-13&0&0&0&-10&0&0&0\\-13&35&-9&0&-11&0&0&0&0\\0&-9&31&-10&0&0&0&0&0\\0&0&-10&79&-30&0&0&0&-9\\0&0&0&-30&57&-7&0&-5&0\\0&0&0&0&-7&47&-30&0&0\\0&0&0&0&0&-30&41&0&0\\0&0&0&0&-5&0&0&27&-2\\0&0&0&-9&0&0&0&-2&29\end{pmatrix}$$

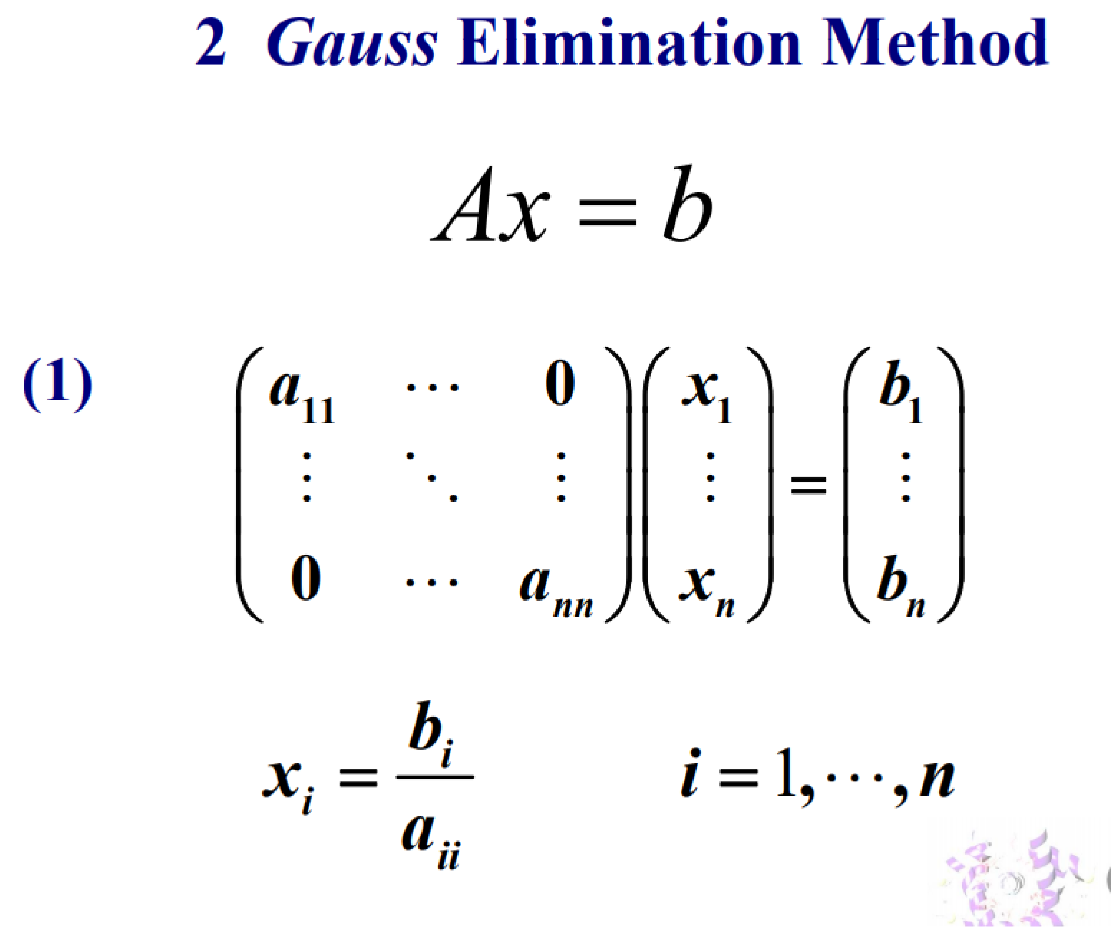


Homework

Write a program to combine the *Gauss-Seidel* and *overrelaxation* method and solve the linear equations.

$$Ax=\begin{pmatrix}-15\\27\\-23\\0\\-20\\12\\-7\\7\\10\end{pmatrix}\qquad A=\begin{pmatrix}31&-13&0&0&0&-10&0&0&0\\-13&35&-9&0&-11&0&0&0&0\\0&-9&31&-10&0&0&0&0&0\\0&0&-10&79&-30&0&0&0&-9\\0&0&0&-30&57&-7&0&-5&0\\0&0&0&0&-7&47&-30&0&0\\0&0&0&0&0&-30&41&0&0\\0&0&0&0&-5&0&0&27&-2\\0&0&0&-9&0&0&0&-2&29\end{pmatrix}$$

- **Formula to use**
  Here we will use four methods in total:
  Gauss Elimination
  Doolittle Decompression
  Gauss-Seidel Iteration
  Overrelaxation Iteration



2 *Gauss* Elimination Method

$$Ax = b$$

(1)

$$\begin{pmatrix} a_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

$$x_i = \frac{b_i}{a_{ii}} \qquad i = 1, \cdots, n$$

*k*th step:

$$row_i + row_k \times m_{ik} = row_i + row_k \times \left(\frac{-a_{ik}^{(k)}}{a_{kk}^{(k)}}\right) \quad i = k+1,\cdots,n$$

After (n-1) steps, we get

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & b_3^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n)} & b_n^{(n)} \end{pmatrix}$$

Get final solution by backward steps

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}} \quad i = 1,\cdots,n$$

Symmetric Positive Definite Matrix

$$A = LL^T$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ & l_{22} & \cdots & l_{n2} \\ & & \ddots & \vdots \\ & & & l_{nn} \end{pmatrix}$$

$$\begin{cases} l_{jj} = (a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2)^{\frac{1}{2}} \quad (j = 1,2,\cdots,n), \\ l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}) / l_{jj} \quad (i = j+1,\cdots,n); \end{cases}$$

2 *Gauss-Seidel* Iteration

$$\begin{cases} x_1^{(k+1)} = \dfrac{-1}{a_{11}}(a_{12}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)} - b_1) \\ x_2^{(k+1)} = \dfrac{-1}{a_{22}}(a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)} + \cdots + a_{1n}x_n^{(k)} - b_2) \\ \qquad \vdots \\ x_n^{(k+1)} = \dfrac{-1}{a_{nn}}(a_{n1}x_1^{(k+1)} + \cdots + a_{n\,n-1}x_{n-1}^{(k+1)} - b_n) \end{cases}$$

$$x_i^{(k+1)} = \frac{-1}{a_{ii}}\left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} - b_i\right)$$

3 Relaxation Iteration

$$x^{(k+1)} = Gx^{(k)} + g$$

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$$

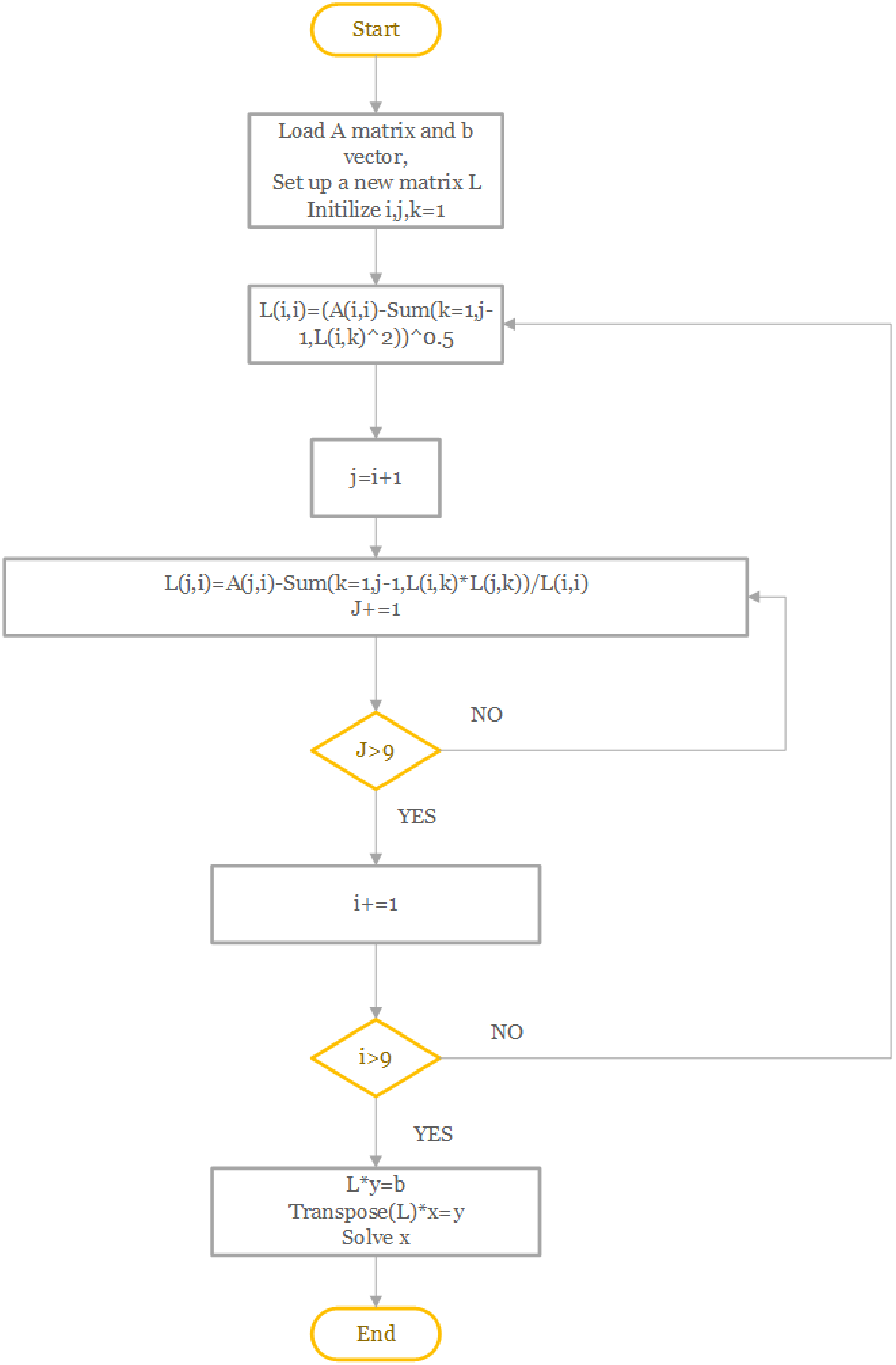
$$\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$$

$$x^{(k+1)} = x^{(k)} + \omega \Delta x^{(k)}$$

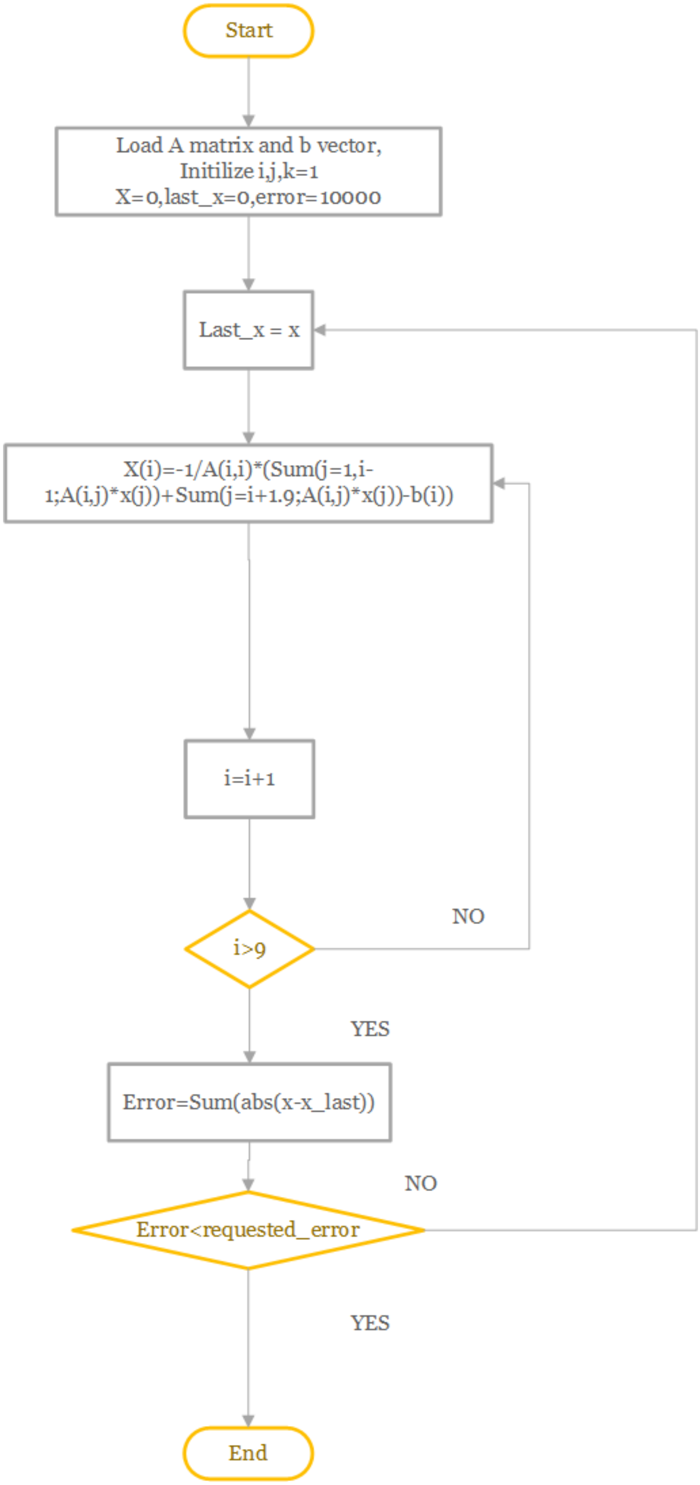
$$x^{(k+1)} = (1-\omega)x^{(k)} + \omega x^{(k+1)}$$

- **Flow chart**

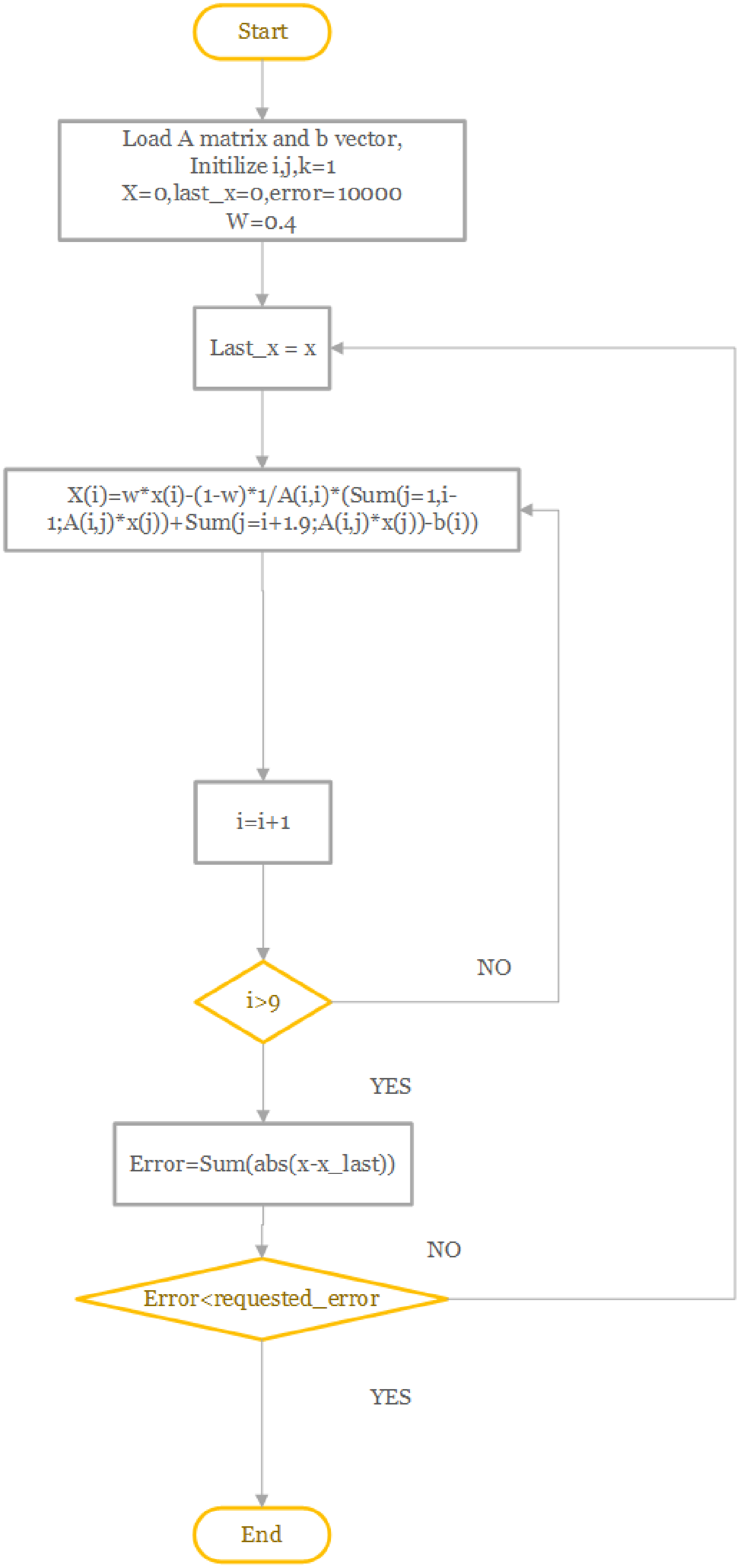
1. **Doolittle Flowchart**


Start
Load A matrix and b vector,
Set up a new matrix L
Initilize i,j,k=1
L(i,i)=(A(i,i)-Sum(k=1,j-1,L(i,k)^2))^0.5
j=i+1
L(j,i)=A(j,i)-Sum(k=1,j-1,L(i,k)*L(j,k))/L(i,i)
J+=1
J>9
NO
YES
i+=1
i>9
NO
YES
L*y=b
Transpose(L)*x=y
Solve x
End

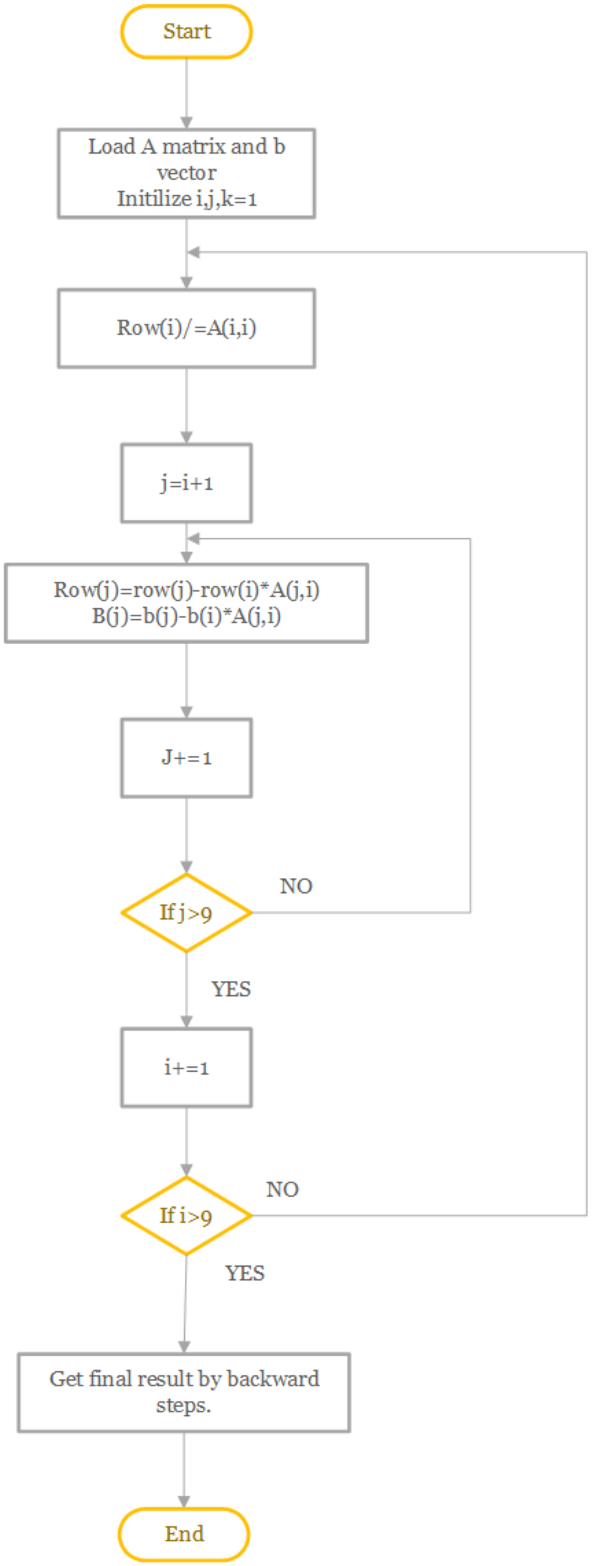
2. Seidel Iteration Flowchart



3. Overrelaxation Iteration Flowchart



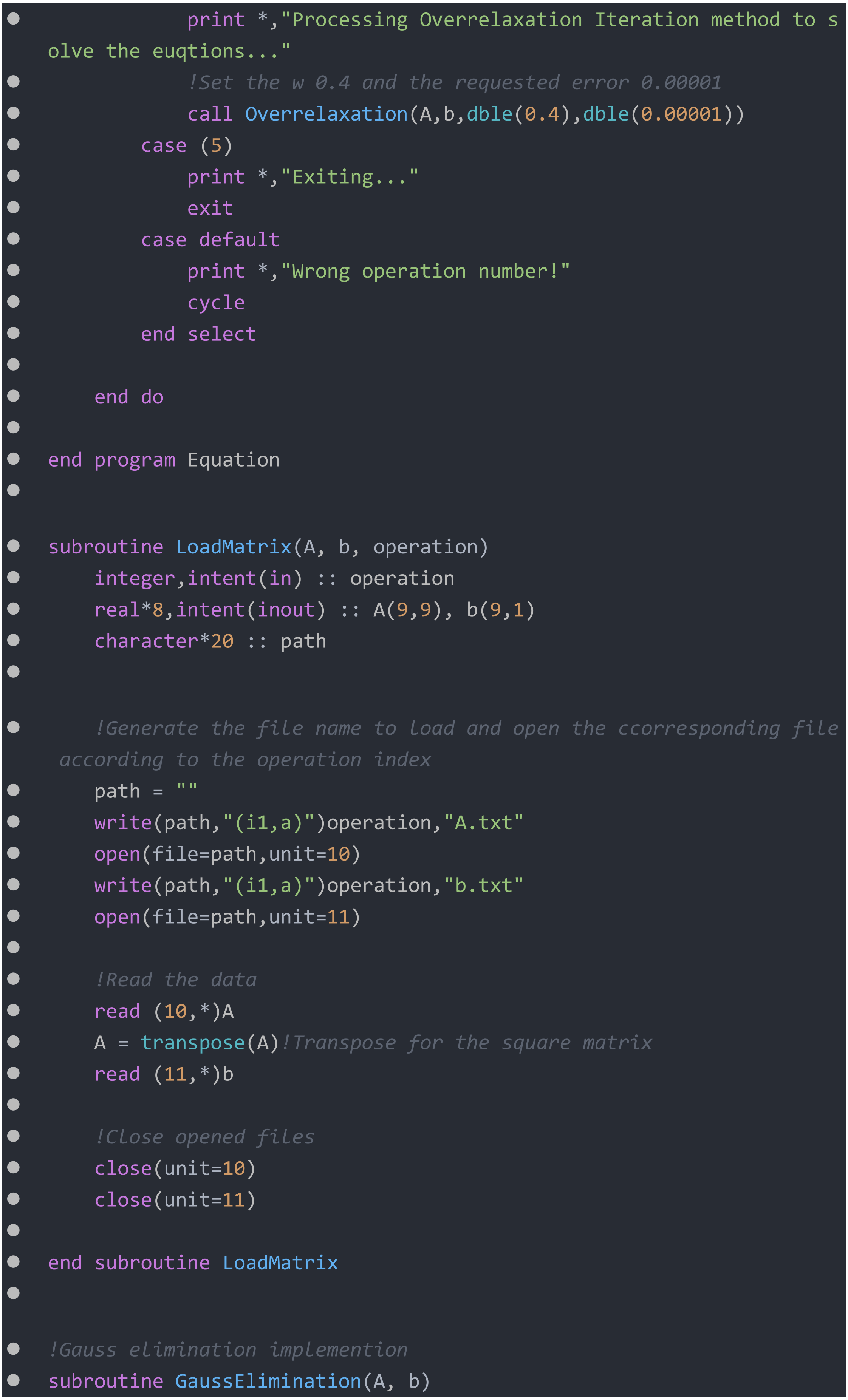
4. Gauss Elimination



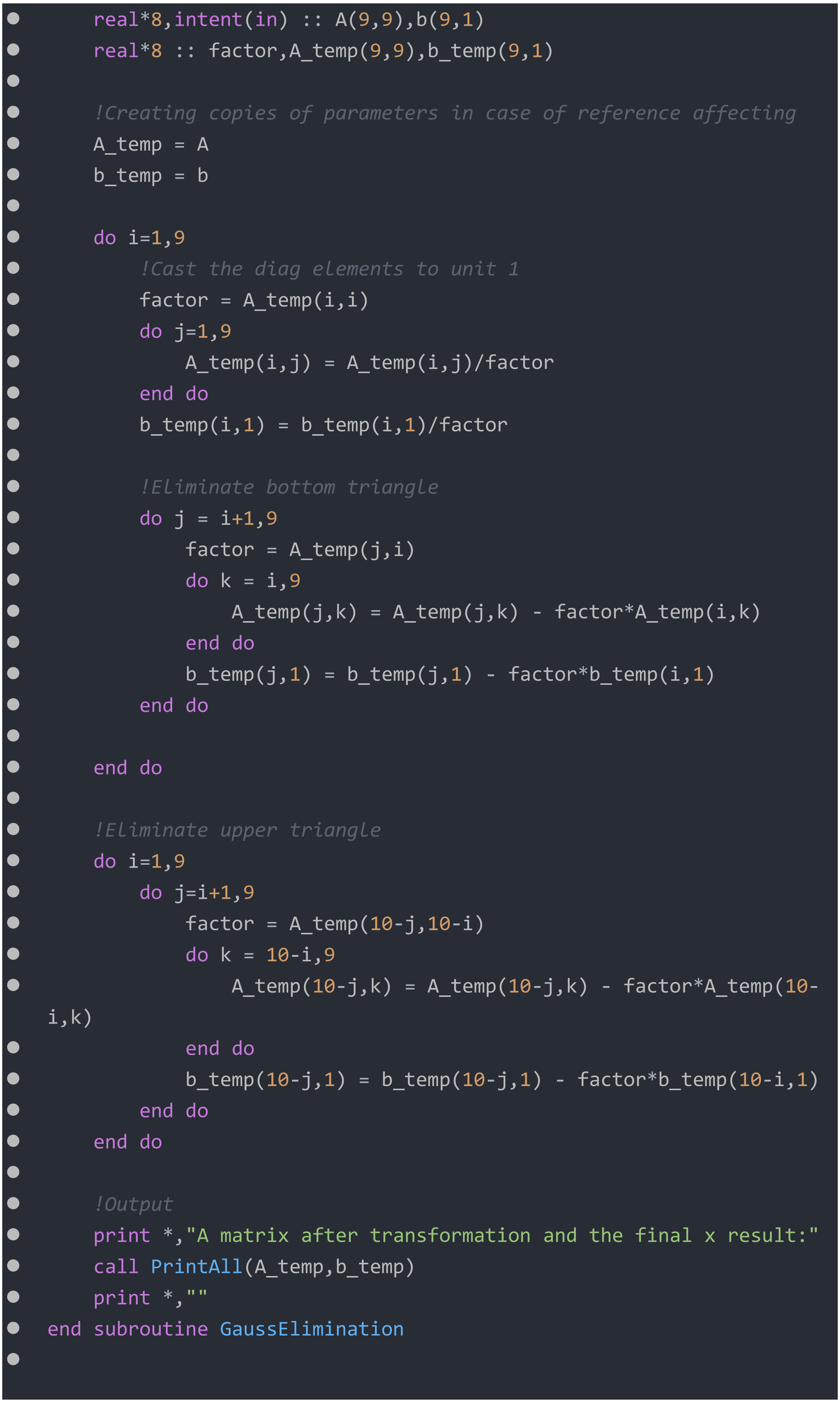
- **Source Code**

```
program Equation
    implicit none
    real*8 :: A(9,9)
```

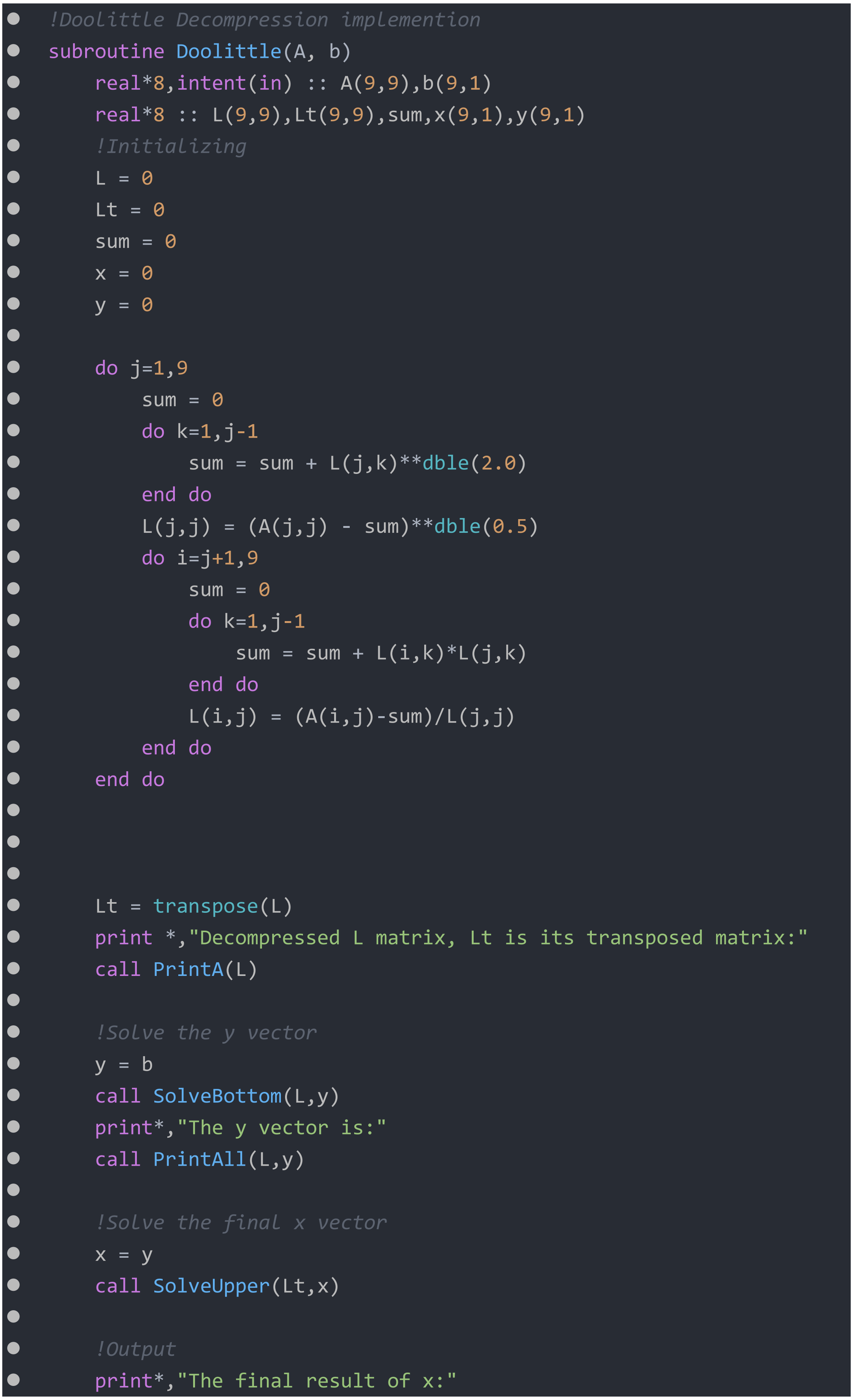
```fortran
    real*8 :: b(9,1)

    integer :: operation
    operation = 0

    do while(.true.)
        print *,"****************************"
        print *,"Enter the operation you would like to choose to solve the equations:"
        print *,"1.Gauss Elimination"
        print *,"2.Doolittle Symetric Decompression"
        print *,"3.Gauss-Seidel Iteration"
        print *,"4.Overrelaxation"
        print *,"5.Exit the programm"
        print *,"If you want to modify the matrix and vector parameter,"
        print *,"please edit the txt file in the directory corresponding to the operation index"
        read *,operation
        !read the operator from the keyboard
        select case(operation)
        case (1)
            print *,"Loading matrix and vector from the file..."
            call LoadMatrix(A,b,1)
            print *,"Processing GaussElimination method to solve the euqtions..."
            call GaussElimination(A,b)

        case (2)
            print *,"Loading matrix and vector from the file..."
            call LoadMatrix(A,b,2)
            print *,"Processing Doolittle Deccompression method to solve the euqtions..."
            call Doolittle(A,b)
        case (3)
            print *,"Loading matrix and vector from the file..."
            call LoadMatrix(A,b,3)
            print *,"Processing Seidel Iteration method to solve the euqtions..."
            !Set the requested error 0.00001
            call Seidel(A,b,dble(0.00001))
        case (4)
            print *,"Loading matrix and vector from the file..."
            call LoadMatrix(A,b,4)
```



```fortran
            print *,"Processing Overrelaxation Iteration method to solve the euqtions..."
            !Set the w 0.4 and the requested error 0.00001
            call Overrelaxation(A,b,dble(0.4),dble(0.00001))
        case (5)
            print *,"Exiting..."
            exit
        case default
            print *,"Wrong operation number!"
            cycle
        end select

    end do

end program Equation


subroutine LoadMatrix(A, b, operation)
    integer,intent(in) :: operation
    real*8,intent(inout) :: A(9,9), b(9,1)
    character*20 :: path


    !Generate the file name to load and open the ccorresponding file according to the operation index
    path = ""
    write(path,"(i1,a)")operation,"A.txt"
    open(file=path,unit=10)
    write(path,"(i1,a)")operation,"b.txt"
    open(file=path,unit=11)

    !Read the data
    read (10,*)A
    A = transpose(A)!Transpose for the square matrix
    read (11,*)b

    !Close opened files
    close(unit=10)
    close(unit=11)

end subroutine LoadMatrix


!Gauss elimination implemention
subroutine GaussElimination(A, b)
```

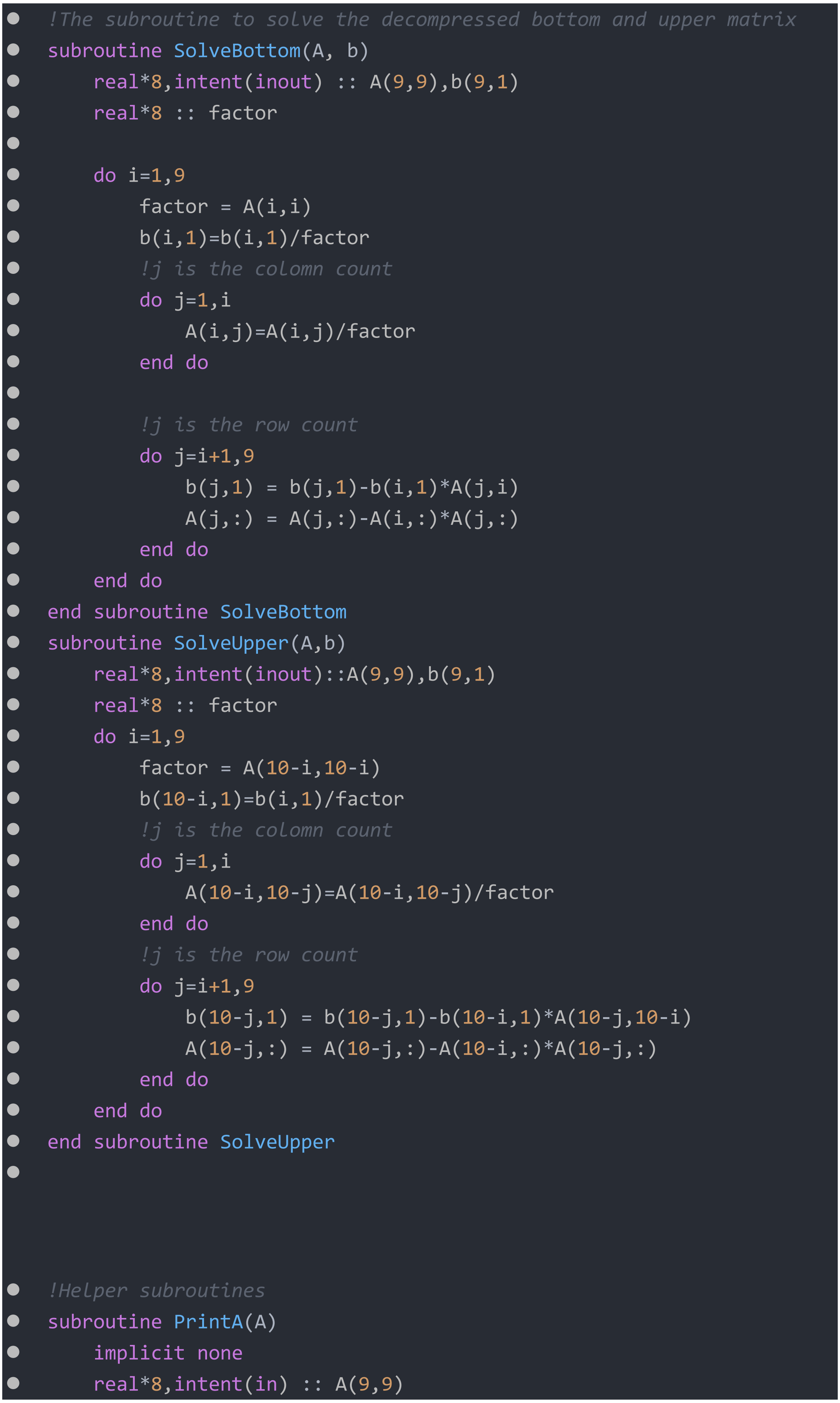


```fortran
    real*8,intent(in) :: A(9,9),b(9,1)
    real*8 :: factor,A_temp(9,9),b_temp(9,1)

    !Creating copies of parameters in case of reference affecting
    A_temp = A
    b_temp = b

    do i=1,9
        !Cast the diag elements to unit 1
        factor = A_temp(i,i)
        do j=1,9
            A_temp(i,j) = A_temp(i,j)/factor
        end do
        b_temp(i,1) = b_temp(i,1)/factor

        !Eliminate bottom triangle
        do j = i+1,9
            factor = A_temp(j,i)
            do k = i,9
                A_temp(j,k) = A_temp(j,k) - factor*A_temp(i,k)
            end do
            b_temp(j,1) = b_temp(j,1) - factor*b_temp(i,1)
        end do

    end do

    !Eliminate upper triangle
    do i=1,9
        do j=i+1,9
            factor = A_temp(10-j,10-i)
            do k = 10-i,9
                A_temp(10-j,k) = A_temp(10-j,k) - factor*A_temp(10-i,k)
            end do
            b_temp(10-j,1) = b_temp(10-j,1) - factor*b_temp(10-i,1)
        end do
    end do

    !Output
    print *,"A matrix after transformation and the final x result:"
    call PrintAll(A_temp,b_temp)
    print *,""
end subroutine GaussElimination

```



```fortran
!Doolittle Decompression implemention
subroutine Doolittle(A, b)
    real*8,intent(in) :: A(9,9),b(9,1)
    real*8 :: L(9,9),Lt(9,9),sum,x(9,1),y(9,1)
    !Initializing
    L = 0
    Lt = 0
    sum = 0
    x = 0
    y = 0

    do j=1,9
        sum = 0
        do k=1,j-1
            sum = sum + L(j,k)**dble(2.0)
        end do
        L(j,j) = (A(j,j) - sum)**dble(0.5)
        do i=j+1,9
            sum = 0
            do k=1,j-1
                sum = sum + L(i,k)*L(j,k)
            end do
            L(i,j) = (A(i,j)-sum)/L(j,j)
        end do
    end do



    Lt = transpose(L)
    print *,"Decompressed L matrix, Lt is its transposed matrix:"
    call PrintA(L)

    !Solve the y vector
    y = b
    call SolveBottom(L,y)
    print*,"The y vector is:"
    call PrintAll(L,y)

    !Solve the final x vector
    x = y
    call SolveUpper(Lt,x)

    !Output
    print*,"The final result of x:"
    call PrintAll(Lt,x)

    print *,""
end subroutine Doolittle

!Seidel Iteration method
subroutine Seidel(A,b,requested_error)
    real*8,intent(in) :: A(9,9),b(9,1),requested_error
    real*8 :: x(9,1),x_last(9,1),sum,error
    integer :: iteration
    !Initialize
    x=0
    x_last=x
    error=100000
    sum=0
    iteration=0

    print *,requested_error
    do while(error>requested_error)
        x_last=x
        do i=1,9
            sum=0
            do j=1,9
                if (j==i)then
                    cycle
                end if
                sum=sum+A(i,j)*x(j,1)
            end do
            x(i,1)=dble(-1)/A(i,i)*(sum-b(i,1))
        end do
        error=0
        do i=1,9
            error=error+abs(x(i,1)-x_last(i,1))
        end do
        iteration=iteration+1
    end do


    !Output
    print *,"The final result of x:"
    call Printb(x)
    print "(a,i4)","Used iteration:",iteration
    print "(a,es10.3)","Final error:",error
end subroutine Seidel
```

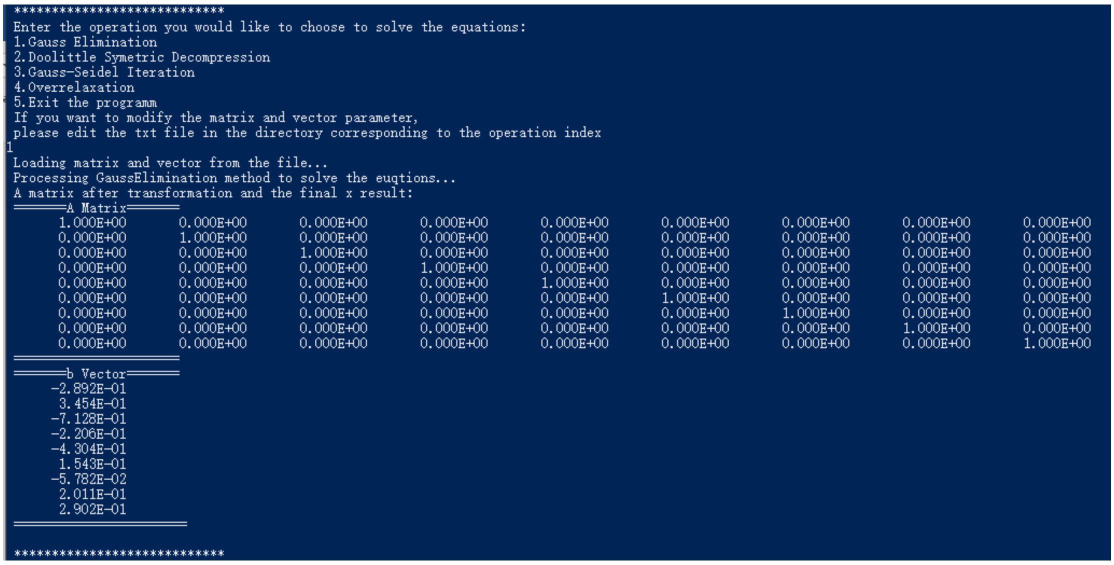
```fortran

!Overrelaxation method implementation
subroutine Overrelaxation(A, b,w ,requested_error)
    real*8,intent(in) :: A(9,9),b(9,1),w,requested_error
    real*8 :: x(9,1),x_last(9,1),sum,error
    integer :: iteration
    !Initialize
    x=0
    x_last=x
    error=100000
    sum=0
    iteration=0

    do while(error>requested_error)
        x_last=x

        do i=1,9
            sum=0
            do j=1,9
                if (j==i)then
                    cycle
                end if
                sum=sum+A(i,j)*x(j,1)
            end do
            x(i,1) = (dble(1)-w)*x(i,1)-w/A(i,i)*(sum-b(i,1))
        end do


        error=0
        do i=1,9
            error=error+abs(x(i,1)-x_last(i,1))
        end do
        iteration=iteration+1
    end do

    !Output
    print *,"The final result of x:"
    call Printb(x)
    print "(a,i4)","Used iteration:",iteration
    print "(a,es10.3)","Final error:",error

end subroutine

```



```
!The subroutine to solve the decompressed bottom and upper matrix
subroutine SolveBottom(A, b)
    real*8,intent(inout) :: A(9,9),b(9,1)
    real*8 :: factor

    do i=1,9
        factor = A(i,i)
        b(i,1)=b(i,1)/factor
        !j is the colomn count
        do j=1,i
            A(i,j)=A(i,j)/factor
        end do

        !j is the row count
        do j=i+1,9
            b(j,1) = b(j,1)-b(i,1)*A(j,i)
            A(j,:) = A(j,:)-A(i,:)*A(j,:)
        end do
    end do
end subroutine SolveBottom
subroutine SolveUpper(A,b)
    real*8,intent(inout)::A(9,9),b(9,1)
    real*8 :: factor
    do i=1,9
        factor = A(10-i,10-i)
        b(10-i,1)=b(i,1)/factor
        !j is the colomn count
        do j=1,i
            A(10-i,10-j)=A(10-i,10-j)/factor
        end do
        !j is the row count
        do j=i+1,9
            b(10-j,1) = b(10-j,1)-b(10-i,1)*A(10-j,10-i)
            A(10-j,:) = A(10-j,:)-A(10-i,:)*A(10-j,:)
        end do
    end do
end subroutine SolveUpper



!Helper subroutines
subroutine PrintA(A)
    implicit none
    real*8,intent(in) :: A(9,9)
    integer :: i
    print *,"=======A Matrix======="
    print "(9es16.3)",(A(i,:),i=1,9)
    print *,"====================="
end subroutine PrintA
subroutine Printb(b)
    implicit none
    real*8,intent(in) :: b(9,1)
    integer :: i
    print *,"=======b Vector======="
    print "(es16.3)",(b(i,:),i=1,9)
    print *,"======================="
end subroutine Printb
subroutine PrintAll(A,b)
    implicit none
    real*8,intent(in) :: A(9,9),b(9,1)
    call PrintA(A)
    call Printb(b)
end subroutine PrintAll
```

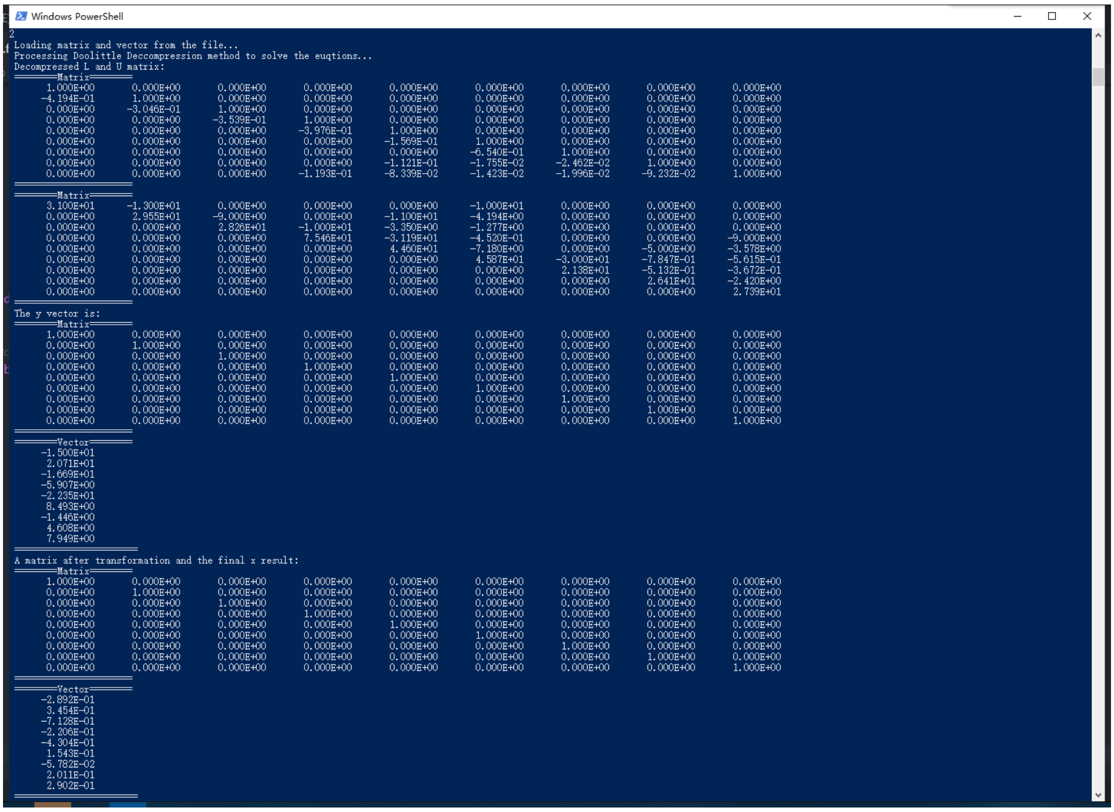
Example and Result

Gauss elimination



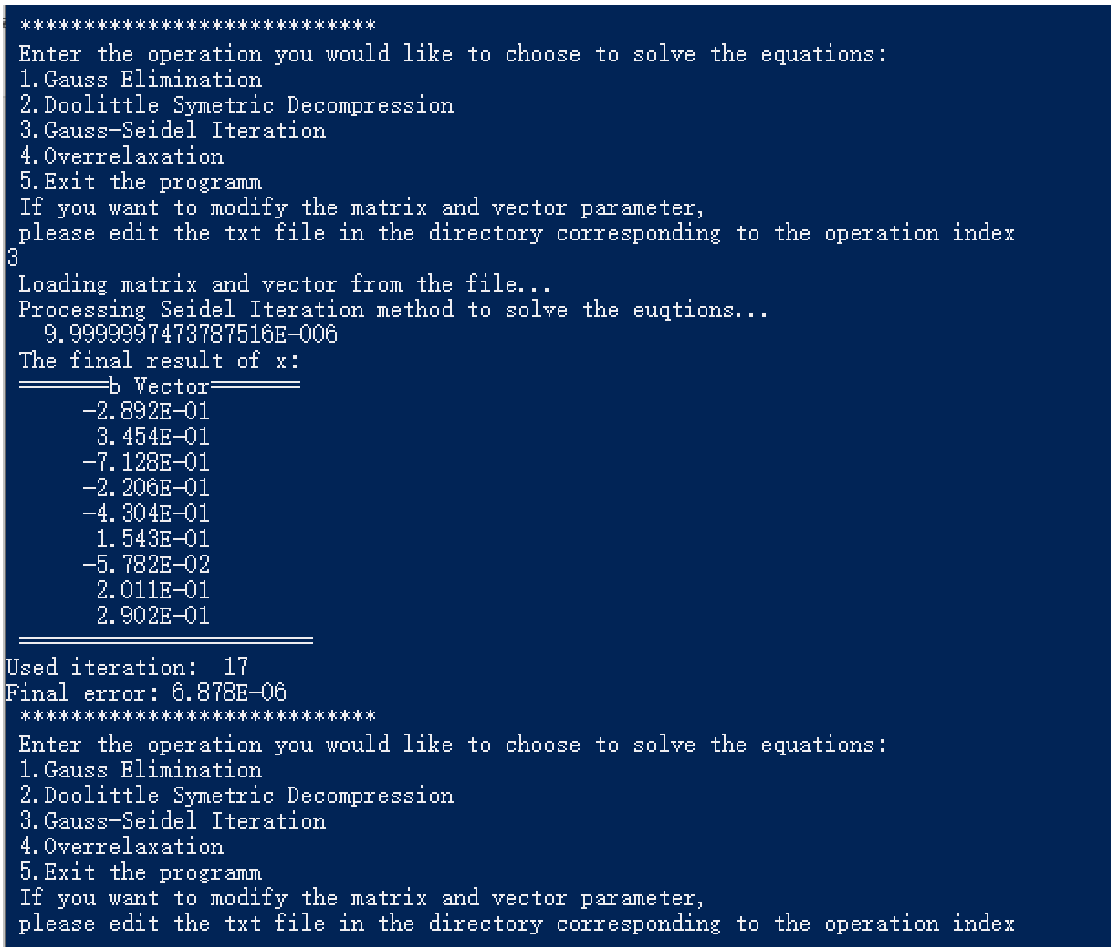
```
 ****************************
 Enter the operation you would like to choose to solve the equations:
 1.Gauss Elimination
 2.Doolittle Symetric Decompression
 3.Gauss-Seidel Iteration
 4.Overrelaxation
 5.Exit the programm
 If you want to modify the matrix and vector parameter,
 please edit the txt file in the directory corresponding to the operation index
1
 Loading matrix and vector from the file...
 Processing GaussElimination method to solve the euqtions...
 A matrix after transformation and the final x result:
 =======A Matrix=======
       1.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00
       0.000E+00       1.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00
       0.000E+00       0.000E+00       1.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00
       0.000E+00       0.000E+00       0.000E+00       1.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00
       0.000E+00       0.000E+00       0.000E+00       0.000E+00       1.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00
       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       1.000E+00       0.000E+00       0.000E+00       0.000E+00
       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       1.000E+00       0.000E+00       0.000E+00
       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       1.000E+00       0.000E+00
       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       0.000E+00       1.000E+00
 =====================

 =======b Vector=======
      -2.892E-01
       3.454E-01
      -7.128E-01
      -2.206E-01
      -4.304E-01
       1.543E-01
      -5.782E-02
       2.011E-01
       2.902E-01
 =======================


 ****************************
```

Doolittle Decompression



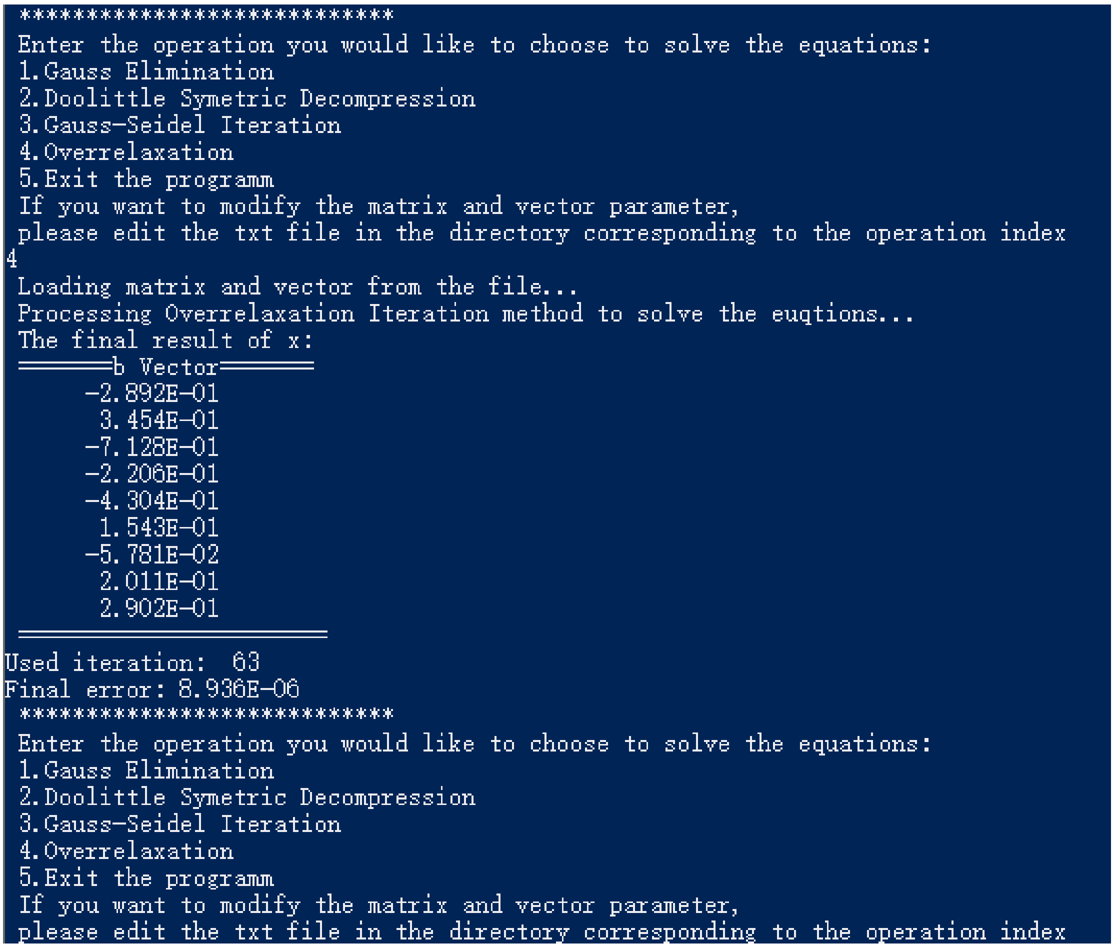
```
2
Loading matrix and vector from the file...
Processing Doolittle Deccompression method to solve the euqtions...
Decompressed L and U matrix:
========Matrix========
    1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
   -4.194E-01     1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00    -3.046E-01     1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00    -3.539E-01     1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00    -3.976E-01     1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00    -1.569E-01     1.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00    -6.540E-01     1.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00    -1.121E-01    -1.755E-02    -2.462E-02     1.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00    -1.193E-01    -8.339E-02    -1.423E-02    -1.996E-02    -9.232E-02     1.000E+00
======================

========Matrix========
    3.100E+01    -1.300E+01     0.000E+00     0.000E+00     0.000E+00    -1.000E+01     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     2.955E+01    -9.000E+00     0.000E+00    -1.100E+01    -4.194E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     2.826E+01    -1.000E+01    -3.350E+00    -1.277E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     7.546E+01    -3.119E+01    -4.520E-01     0.000E+00     0.000E+00    -9.000E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     4.460E+01    -7.180E+00     0.000E+00    -5.000E+00    -3.578E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     4.587E+01    -3.000E+01    -7.847E-01    -5.615E-01
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     2.138E+01    -5.132E-01    -3.672E-01
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     2.641E+01    -2.420E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     2.739E+01
======================

The y vector is:
========Matrix========
    1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     1.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     1.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     1.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     1.000E+00
======================

========Vector========
   -1.500E+01
    2.071E+01
   -1.669E+01
   -5.907E+00
   -2.235E+01
    8.493E+00
   -1.446E+00
    4.608E+00
    7.949E+00
======================

A matrix after transformation and the final x result:
========Matrix========
    1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     1.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     1.000E+00     0.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     1.000E+00     0.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     1.000E+00     0.000E+00
    0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00     1.000E+00
======================

========Vector========
   -2.892E-01
    3.454E-01
   -7.128E-01
   -2.206E-01
   -4.304E-01
    1.543E-01
   -5.782E-02
    2.011E-01
    2.902E-01
======================
```

■ Gauss-Seidel Iteration



```
 ****************************
 Enter the operation you would like to choose to solve the equations:
 1.Gauss Elimination
 2.Doolittle Symetric Decompression
 3.Gauss-Seidel Iteration
 4.Overrelaxation
 5.Exit the programm
 If you want to modify the matrix and vector parameter,
 please edit the txt file in the directory corresponding to the operation index
3
 Loading matrix and vector from the file...
 Processing Seidel Iteration method to solve the euqtions...
   9.9999997473787516E-006
 The final result of x:
 =======b Vector=======
      -2.892E-01
       3.454E-01
      -7.128E-01
      -2.206E-01
      -4.304E-01
       1.543E-01
      -5.782E-02
       2.011E-01
       2.902E-01
 =======================

Used iteration:  17
Final error: 6.878E-06
 ****************************
 Enter the operation you would like to choose to solve the equations:
 1.Gauss Elimination
 2.Doolittle Symetric Decompression
 3.Gauss-Seidel Iteration
 4.Overrelaxation
 5.Exit the programm
 If you want to modify the matrix and vector parameter,
 please edit the txt file in the directory corresponding to the operation index
```

- ### Overrelaxation



```
 *****************************
 Enter the operation you would like to choose to solve the equations:
 1.Gauss Elimination
 2.Doolittle Symetric Decompression
 3.Gauss-Seidel Iteration
 4.Overrelaxation
 5.Exit the programm
 If you want to modify the matrix and vector parameter,
 please edit the txt file in the directory corresponding to the operation index
4
 Loading matrix and vector from the file...
 Processing Overrelaxation Iteration method to solve the euqtions...
 The final result of x:
 =======b Vector=======
      -2.892E-01
       3.454E-01
      -7.128E-01
      -2.206E-01
      -4.304E-01
       1.543E-01
      -5.781E-02
       2.011E-01
       2.902E-01
 =======================

Used iteration:  63
Final error: 8.936E-06
 *****************************
 Enter the operation you would like to choose to solve the equations:
 1.Gauss Elimination
 2.Doolittle Symetric Decompression
 3.Gauss-Seidel Iteration
 4.Overrelaxation
 5.Exit the programm
 If you want to modify the matrix and vector parameter,
 please edit the txt file in the directory corresponding to the operation index
```

- **Demo**

  Check the folder ”Equations” in the directory and follow the instruction to set up the matrices and vectors