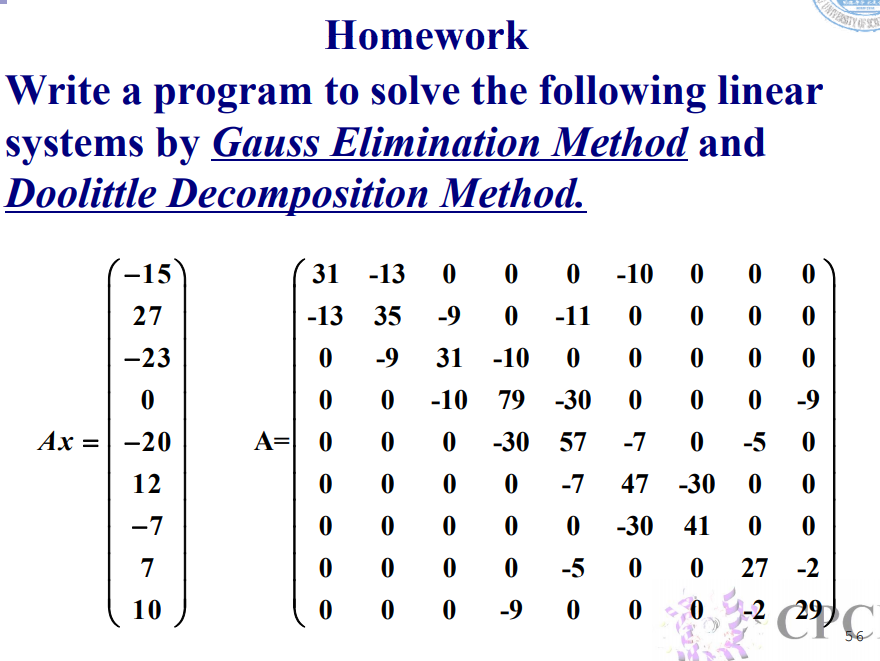
# GThe third schoolwork of Computational Physics

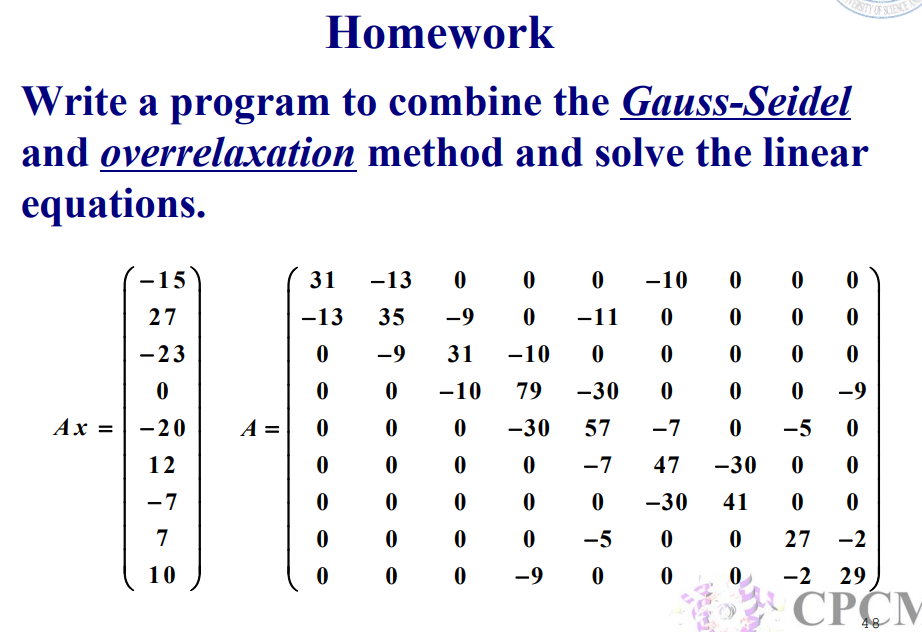
万炫均 物理1701 U201710170

**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**





* **Formula to use**

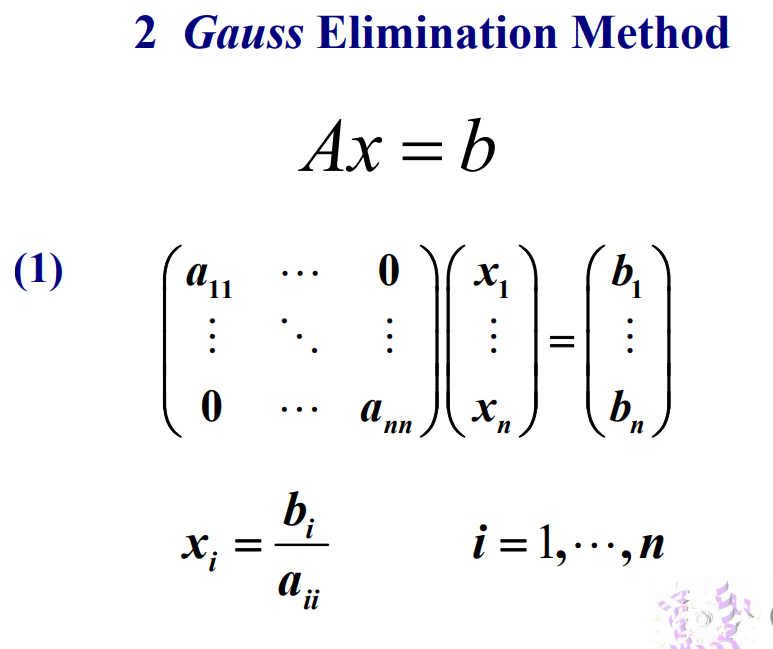
Here we will use four methods in total:

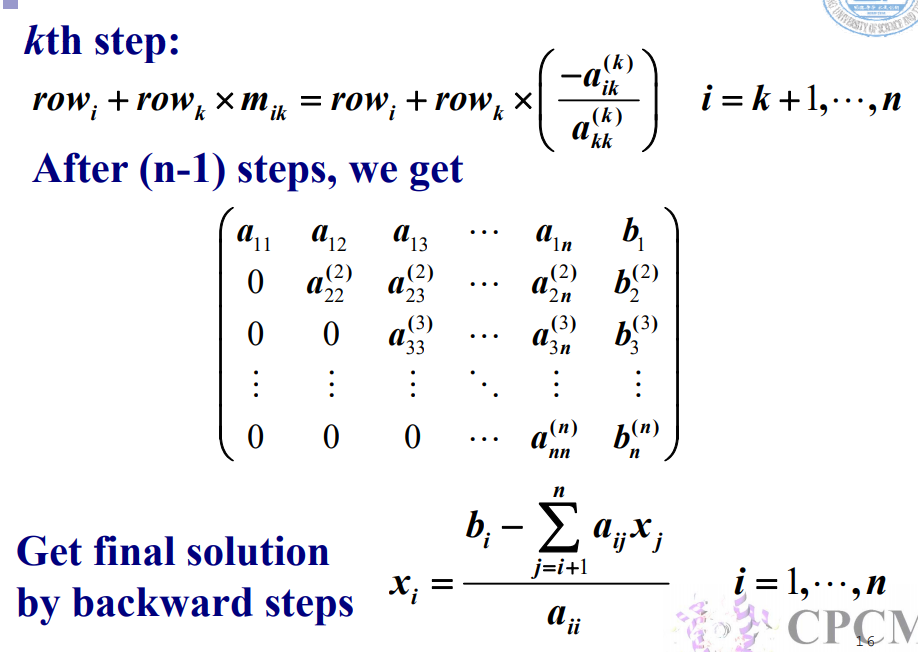
Gauss Elimination

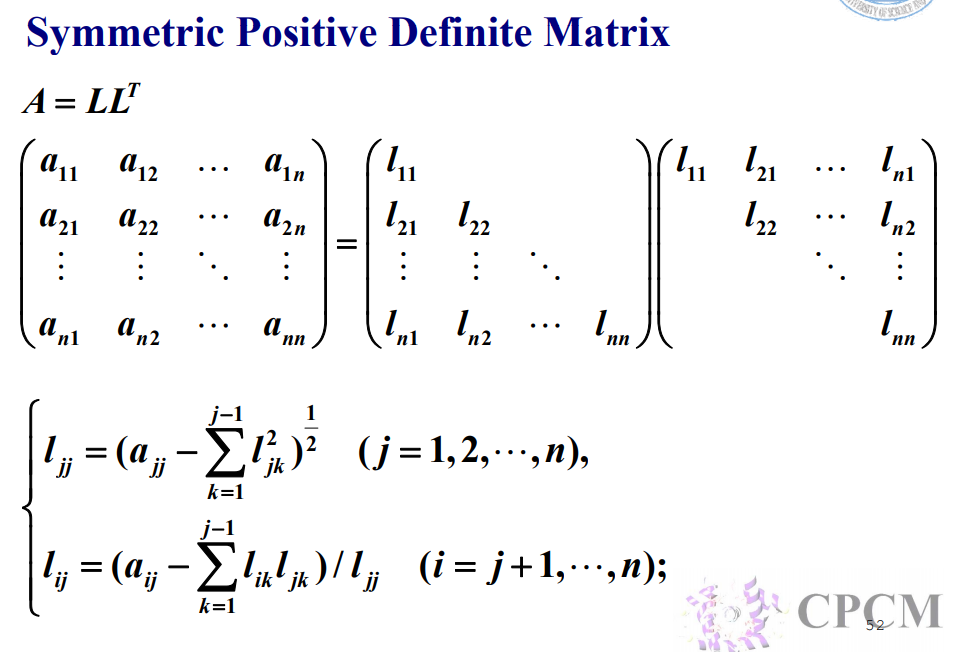
Doolittle Decompression

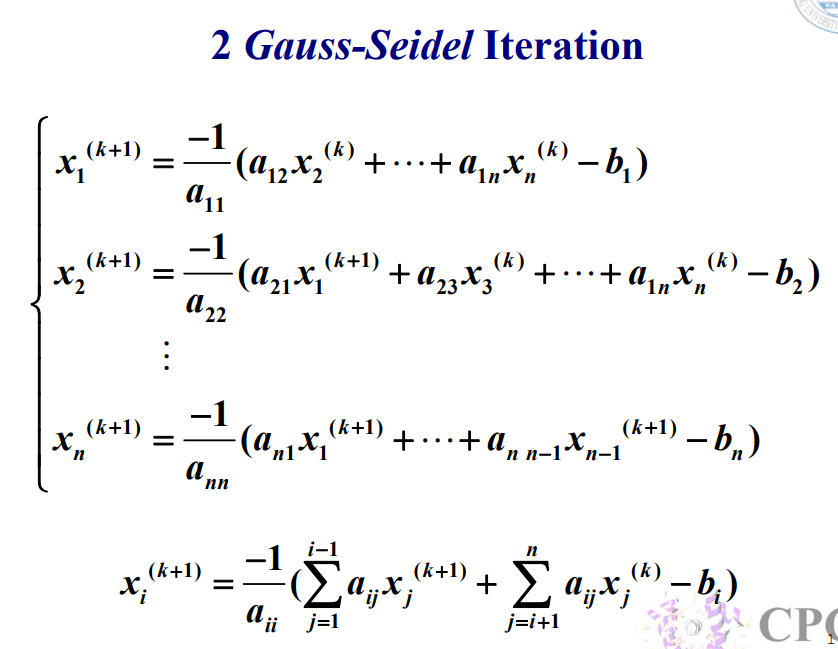
Gauss-Seidel Iteration

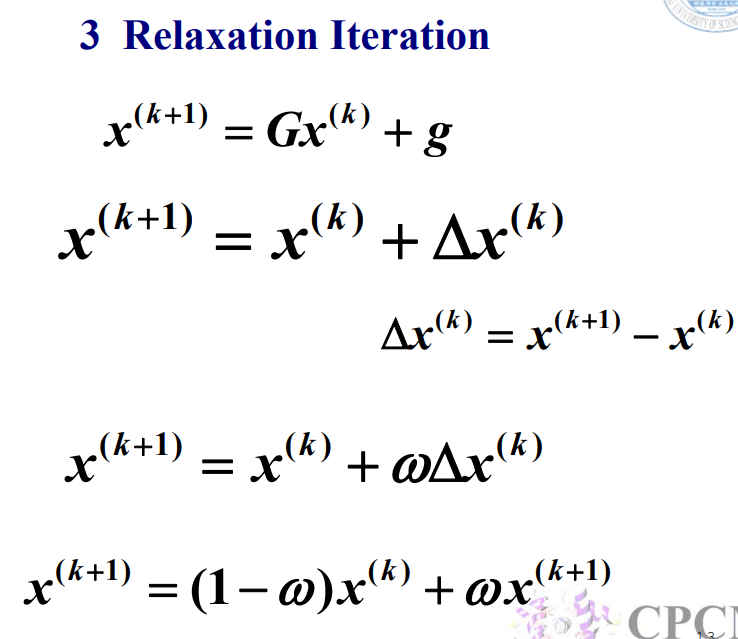
Overrelaxation Iteration





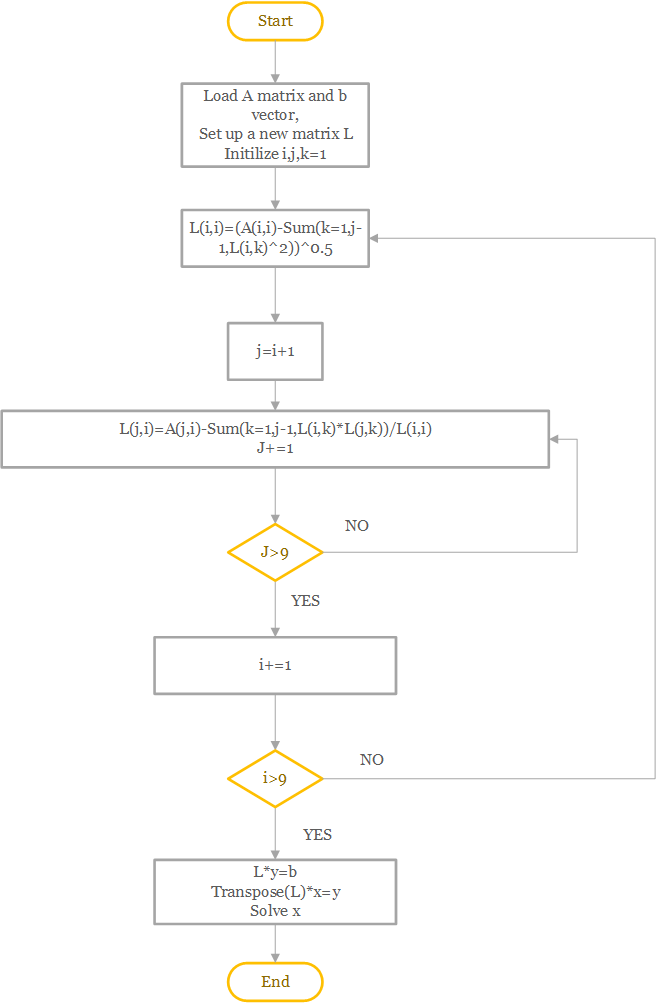




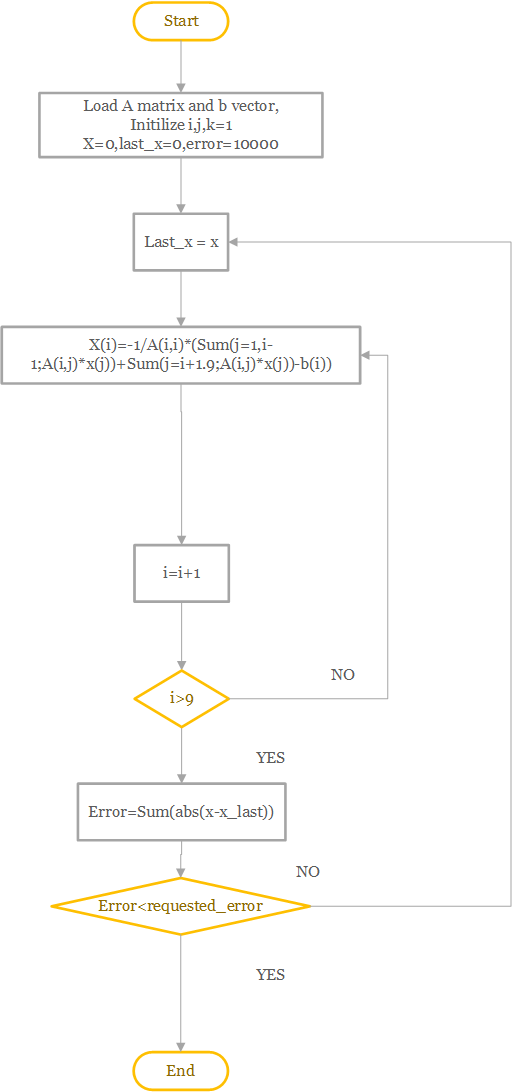


* **Flow chart**

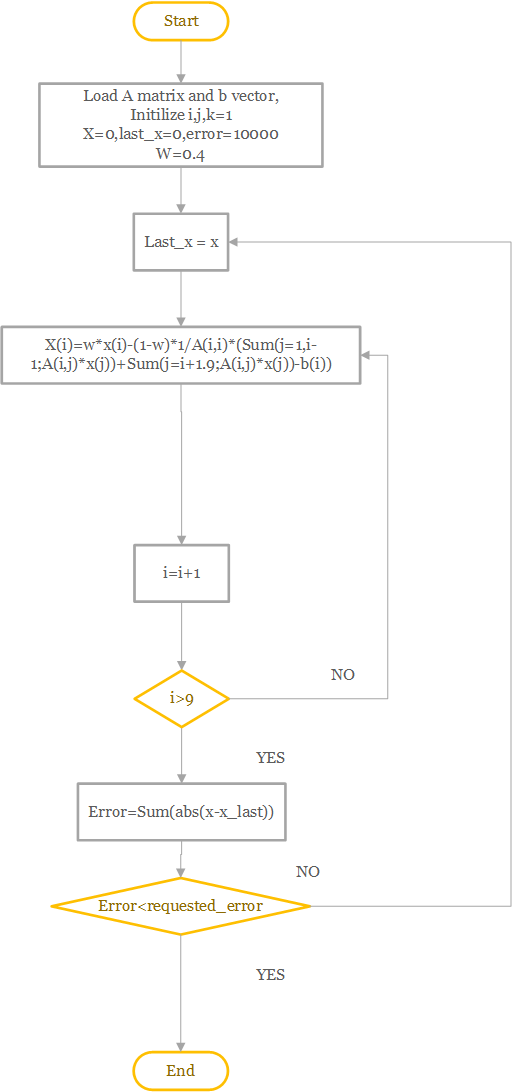
1. **Doolittle Flowchart**

****

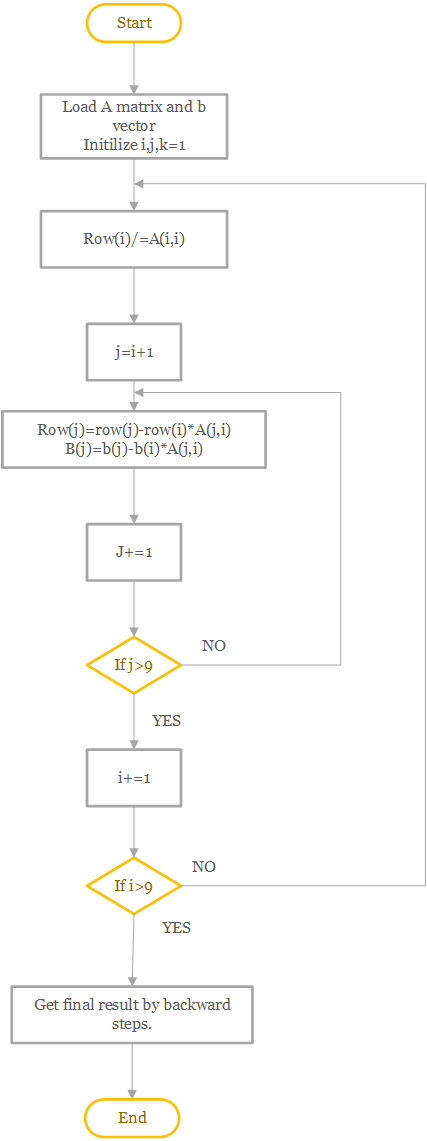
1. **Seidel Iteration Flowchart**

****

1. **Overrelaxation Iteration Flowchart**

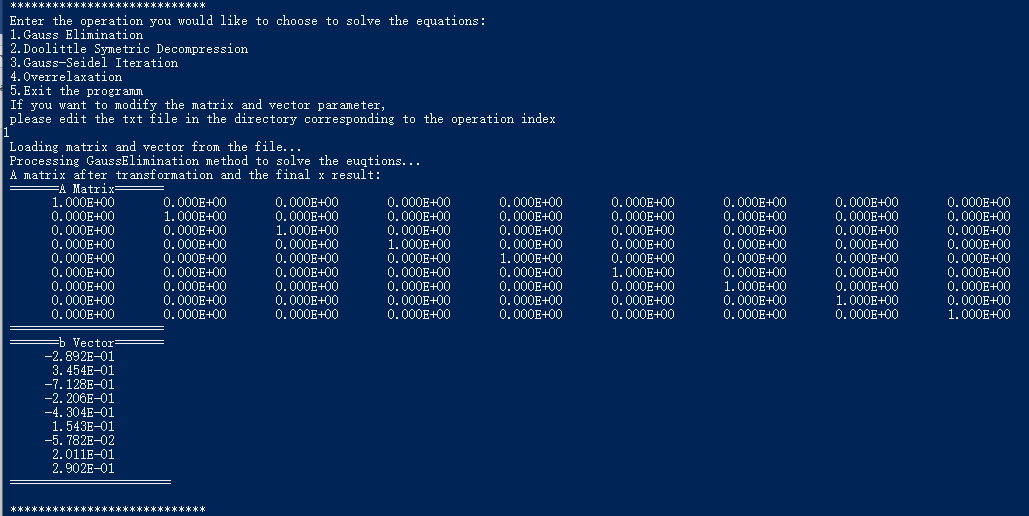
****

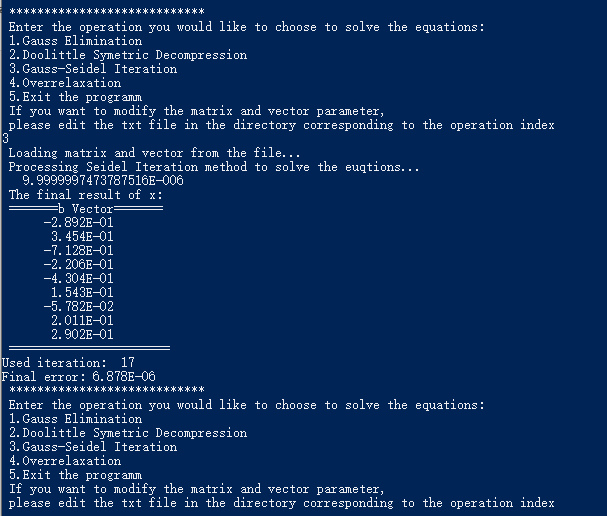
1. **Gauss Elimination**

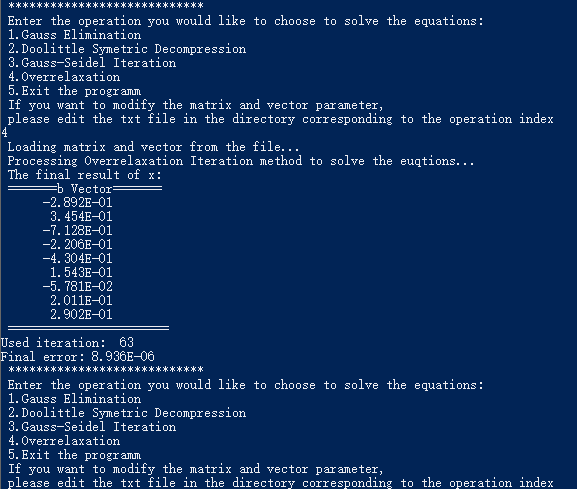
****

* **Source Code**
* program Equation
* implicit none
* real\*8 :: A(9,9)
* 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)
* 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)
* 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
* *!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
* *!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**







* **Demo**

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