Linear Equation Solver

报告主要内容

Write a program to solve the following linear systems by Gauss Elimination Method, Doolittle Decomposition Method and combine the Gauss-Seidel and overrelaxation method.

$$Ax = \begin{pmatrix} -15 \\ 27 \\ -23 \\ 0 \\ -20 \\ 12 \\ -7 \\ 7 \\ 10 \end{pmatrix} \tag{168}$$

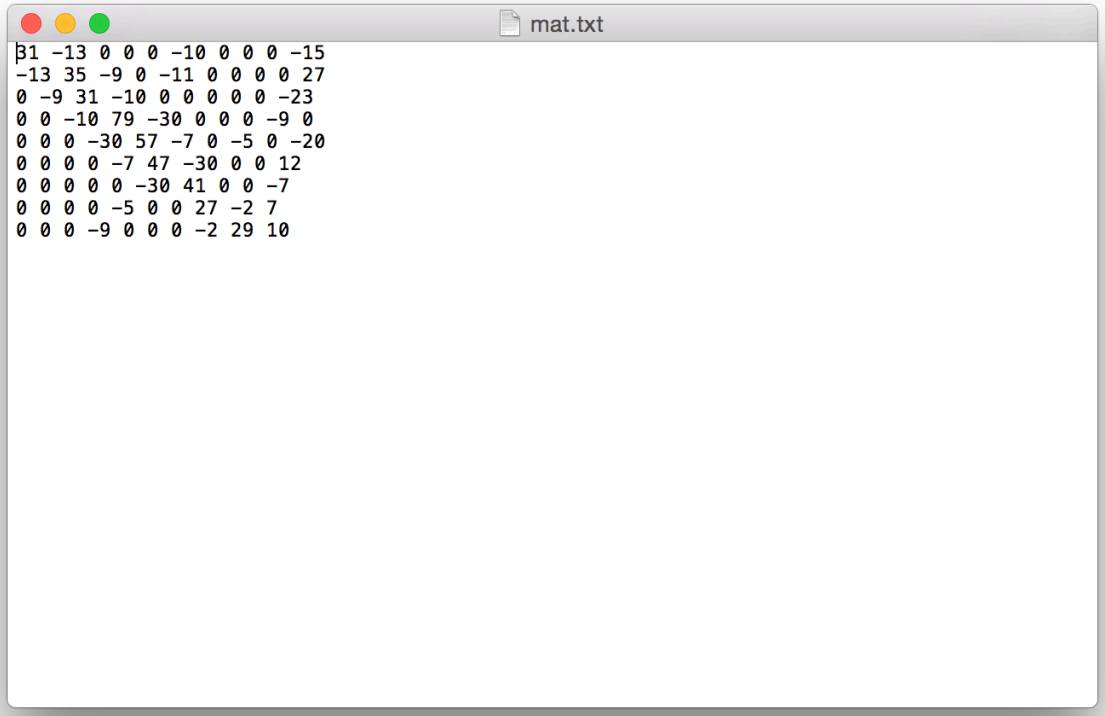
$$A = \begin{pmatrix} 31 & -13 & 0 & 0 & 0 & -10 & 0 & 0 & 0 \\ -13 & 35 & -9 & 0 & -11 & 0 & 0 & 0 & 0 \\ 0 & -9 & 31 & -10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -10 & 79 & -30 & 0 & 0 & 0 & -9 \\ 0 & 0 & 0 & -30 & 57 & -7 & 0 & -5 & 0 \\ 0 & 0 & 0 & 0 & -7 & 47 & -30 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -30 & 41 & 0 & 0 \\ 0 & 0 & 0 & 0 & -5 & 0 & 0 & 27 & -2 \\ 0 & 0 & 0 & -9 & 0 & 0 & 0 & -2 & 29 \end{pmatrix} \tag{169}$$

Main program

```fortran
program main
implicit none
real,dimension(9,10) :: matrix
integer :: i,j

! Read data from file
open(10,file='mat.txt')
read(10,*) ((matrix(i,j),j=1,10),i=1,9)

! Start
print *
print *,'==============='
print *,'The matrix is:'
print *,'==============='
print *
call printMat(matrix)
print *
call gauss_elimination(matrix)
call doolittle_decomposition(matrix)
call gauss_seidel(matrix)
call over_relaxation(matrix)

end program main
```

调用四个子程序

`gauss_elimination`, `doolittle_decomposition`, `gauss_seidel`, `over_relaxation`, 数据矩阵储存在 `mat.txt` 中。



下面是一些要用到的方法:

1.打印数据矩阵到屏幕

```fortran
! ==========================
! Print the matrix on screen
subroutine printMat(m)
implicit none
real,dimension(9,10) :: m
integer :: i,j,a,b
a=size(m(:,1))
b=size(m(1,:))
write(*,"(f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4)")
((m(i,j),j=1,b),i=1,a)
end subroutine printMat

subroutine printMat2(m)
implicit none
real,dimension(9,9) :: m
integer :: i,j,a,b
a=size(m(:,1))
b=size(m(1,:))
write(*,"(f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4)")
((m(i,j),j=1,b),i=1,a)
end subroutine printMat2
```

2.一些基本的矩阵操作

```
! ============================
! Change row i and j of matrix
subroutine change_row(m,i,j)
implicit none
real,dimension(9,10) :: m
real,dimension(10) :: t
integer :: i,j
t=m(i,:)
m(i,:)=m(j,:)
m(j,:)=t
end subroutine change_row

! ============================
! Exchange Column Pivot Element
! c: Column
subroutine column_pivot(m,c)
implicit none
real,dimension(9,10) :: m
real :: t
integer :: i,j,c
t=0.0
j=c
do i=c,size(m(:,1))
    if (abs(m(i,c)) > t) then
        t=abs(m(i,c))
        j=i
    end if
end do
call change_row(m,c,j)
end subroutine column_pivot

! ================================
! Row i=i+c*j
subroutine row_i_plus_Cxj(m,i,j,c)
implicit none
real,dimension(9,10) :: m
real :: c
integer :: i,j
m(i,:)=m(i,:)+c*m(j,:)
end subroutine row_i_plus_Cxj
```

3.上下三角矩阵求解

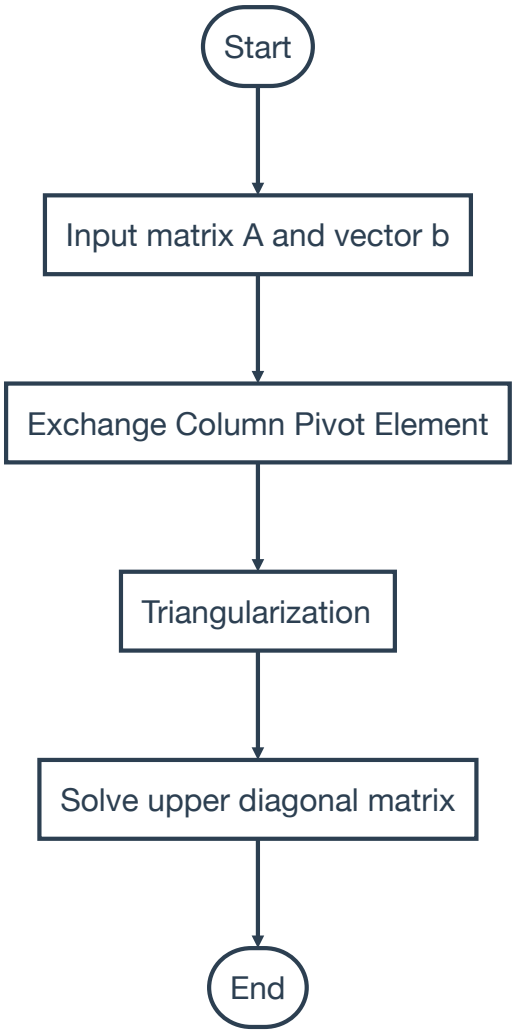
```
! ========================================================
! Triangularization
! - Transform augmented matrix into upper triangular matrix
```

```fortran
subroutine triangularization(m)
implicit none
real,dimension(9,10) :: m
integer :: i,j,n
n=size(m(:,1))

do j=1,n-1
    call column_pivot(m,j)
    do i=j+1,n
        call row_i_plus_Cxj(m,i,j,-m(i,j)/m(j,j))
    end do
end do

end subroutine triangularization

! ============================
! Solve upper diagonal matrix
subroutine solve_upper_diagonal(m,x)
real,dimension(9,10) :: m
real,dimension(9) :: x
real :: s
integer :: n,i,j
n=size(m(:,1))

x(n)=m(n,n+1)/m(n,n)
do i=n-1,1,-1
    s=0.0
    do j=i+1,n
        s=s+m(i,j)*x(j)
    end do
    x(i)=(m(i,n+1)-s)/m(i,i)
end do

end subroutine solve_upper_diagonal

! ============================
! Solve lower diagonal matrix
subroutine solve_lower_diagonal(m,x)
real,dimension(9,10) :: m
real,dimension(9) :: x
real :: s
integer :: n,i,j
n=size(m(:,1))

x(n)=m(n,n+1)/m(n,n)
do i=1,n
    s=0.0
    do j=1,i-1
        s=s+m(i,j)*x(j)
```

```
        end do
        x(i)=(m(i,n+1)-s)/m(i,i)
    end do

    end subroutine solve_lower_diagonal
```

Gauss Elimination Method

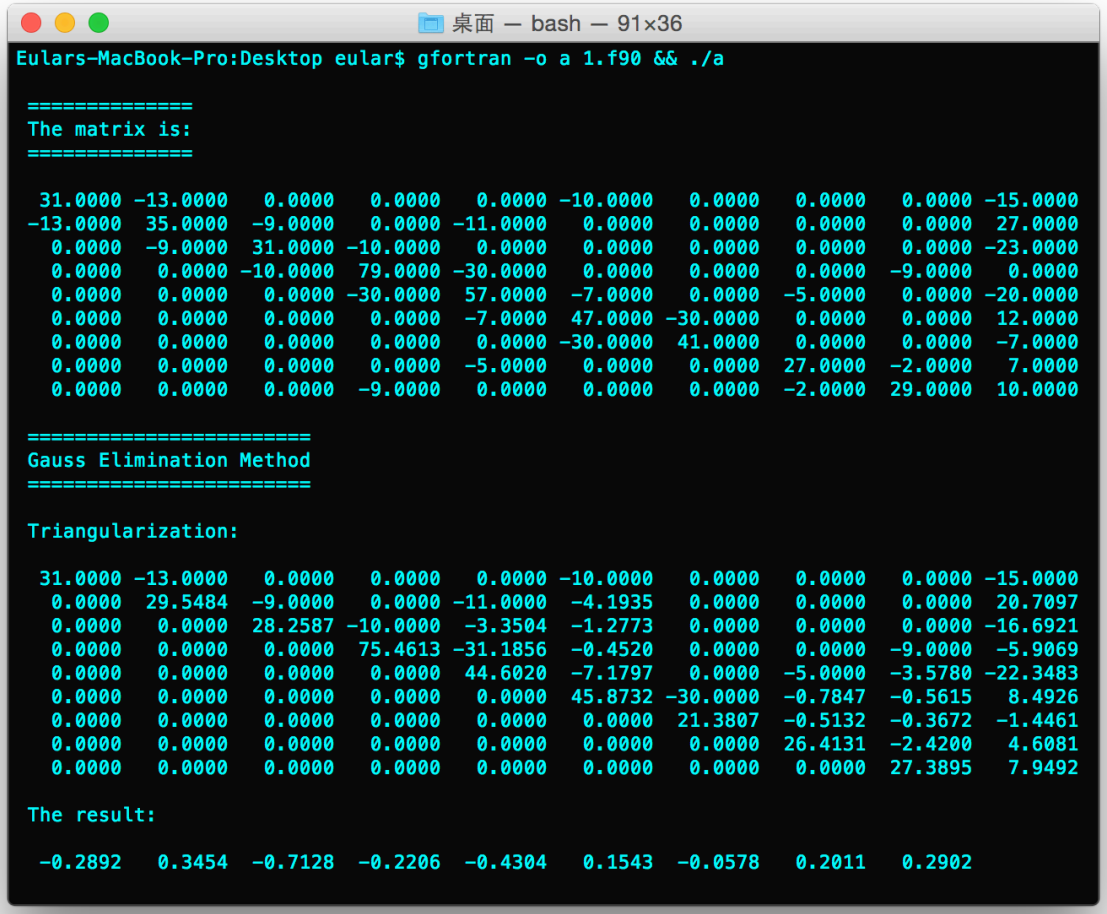
流程图



Code

```fortran
! ====================================
! Gauss Elimination Method
subroutine gauss_elimination(matrix)
implicit none
real,dimension(9,10) :: m,matrix
real,dimension(9) :: x=0
integer :: i,j
m=matrix

print *,'========================'
print *,'Gauss Elimination Method'
print *,'========================'
print *
print *,'Triangularization:'
print *
call triangularization(m)
call printMat(m)
print *
print *,'The result:'
print *
call solve_upper_diagonal(m,x)
write(*,'(f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4)') x
print *

end subroutine gauss_elimination
```

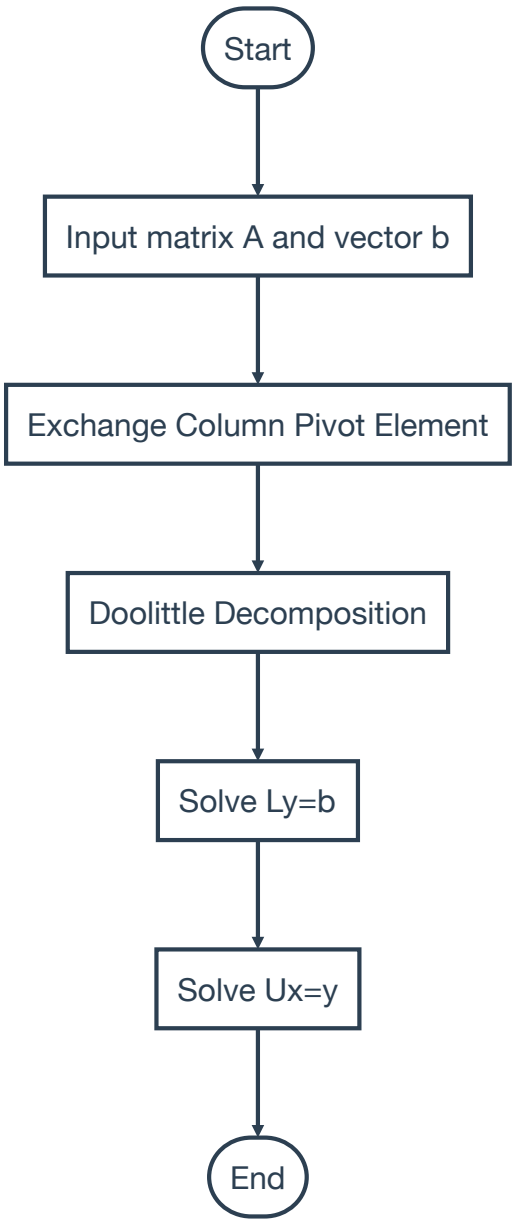
Result



```
桌面 — bash — 91×36
Eulars-MacBook-Pro:Desktop eular$ gfortran -o a 1.f90 && ./a

 ==============
 The matrix is:
 ==============

  31.0000 -13.0000   0.0000   0.0000   0.0000 -10.0000   0.0000   0.0000   0.0000 -15.0000
 -13.0000  35.0000  -9.0000   0.0000 -11.0000   0.0000   0.0000   0.0000   0.0000  27.0000
   0.0000  -9.0000  31.0000 -10.0000   0.0000   0.0000   0.0000   0.0000   0.0000 -23.0000
   0.0000   0.0000 -10.0000  79.0000 -30.0000   0.0000   0.0000   0.0000  -9.0000   0.0000
   0.0000   0.0000   0.0000 -30.0000  57.0000  -7.0000   0.0000  -5.0000   0.0000 -20.0000
   0.0000   0.0000   0.0000   0.0000  -7.0000  47.0000 -30.0000   0.0000   0.0000  12.0000
   0.0000   0.0000   0.0000   0.0000   0.0000 -30.0000  41.0000   0.0000   0.0000  -7.0000
   0.0000   0.0000   0.0000   0.0000  -5.0000   0.0000   0.0000  27.0000  -2.0000   7.0000
   0.0000   0.0000   0.0000  -9.0000   0.0000   0.0000   0.0000  -2.0000  29.0000  10.0000

 ========================
 Gauss Elimination Method
 ========================

 Triangularization:

  31.0000 -13.0000   0.0000   0.0000   0.0000 -10.0000   0.0000   0.0000   0.0000 -15.0000
   0.0000  29.5484  -9.0000   0.0000 -11.0000  -4.1935   0.0000   0.0000   0.0000  20.7097
   0.0000   0.0000  28.2587 -10.0000  -3.3504  -1.2773   0.0000   0.0000   0.0000 -16.6921
   0.0000   0.0000   0.0000  75.4613 -31.1856  -0.4520   0.0000   0.0000  -9.0000  -5.9069
   0.0000   0.0000   0.0000   0.0000  44.6020  -7.1797   0.0000  -5.0000  -3.5780 -22.3483
   0.0000   0.0000   0.0000   0.0000   0.0000  45.8732 -30.0000  -0.7847  -0.5615   8.4926
   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000  21.3807  -0.5132  -0.3672  -1.4461
   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000  26.4131  -2.4200   4.6081
   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000  27.3895   7.9492

 The result:

  -0.2892   0.3454  -0.7128  -0.2206  -0.4304   0.1543  -0.0578   0.2011   0.2902
```

Doolittle Decomposition Method

流程图



Code

```
! ========================================
! Doolittle Decomposition Method
subroutine doolittle_decomposition(matrix)
implicit none
real,dimension(9,10) :: m,matrix
real,dimension(9,9) :: L=0,U=0
real,dimension(9) :: x=0,y=0
integer :: i,j,n
m=matrix
n=size(m(:,1))

print *,'==============================='
print *,'Doolittle Decomposition Method'
print *,'==============================='
print *
```

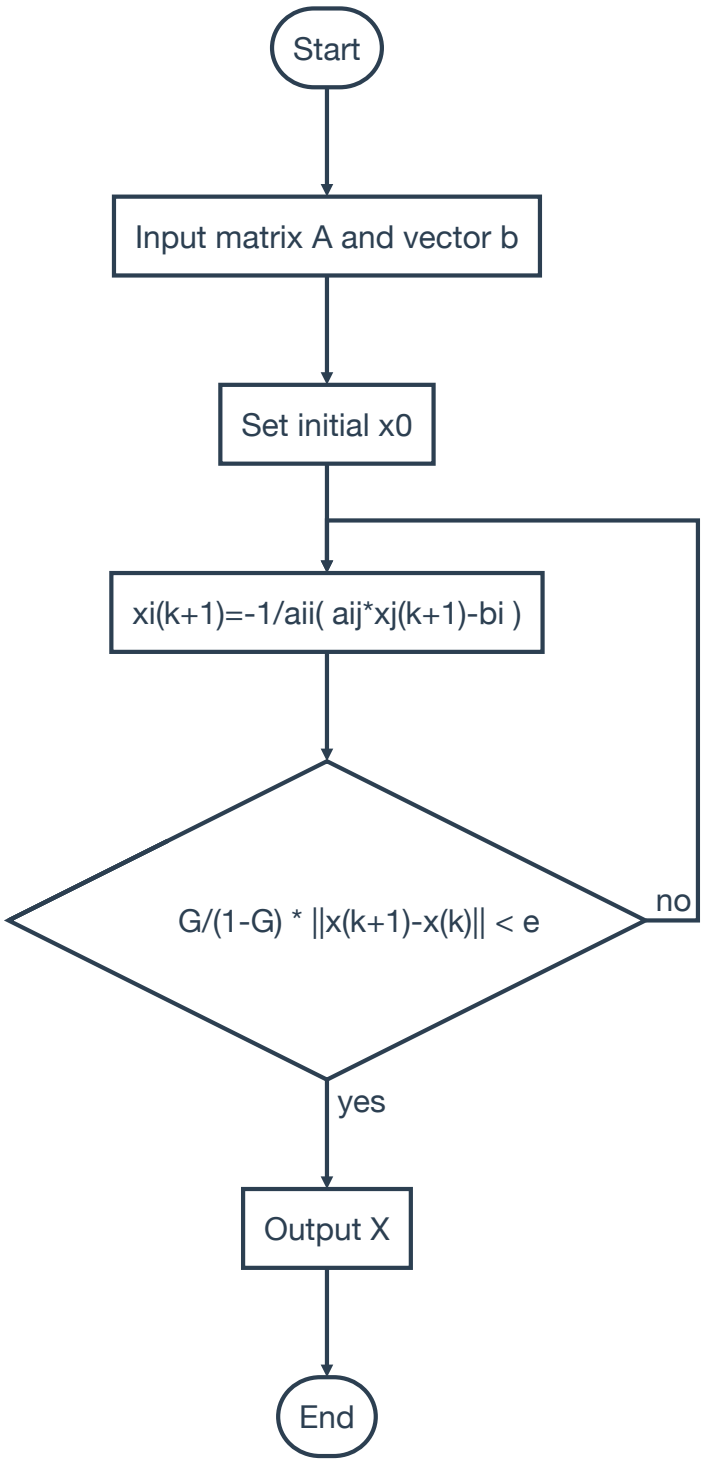
```
    print *,'Exchange Column Pivot Element'
    print *
    do i=1,n-1
        call column_pivot(m,i)
    end do
    call printMat(m)
    print *
    print *,'LU Decomposition'
    print *
    call doolittle(m,L,U)
    print *,'* lower triangular matrix L'
    print *
    call printMat2(L)
    print *
    print *,'* upper triangular matrix U'
    print *
    call printMat2(U)
    print *
    print *,'Solve Ly=b, get y'
    print *
    m(:n,:n)=L
    call solve_lower_diagonal(m,y)
    write(*,'(f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4)') y
    print *
    print *,'Solve Ux=y, get x'
    print *
    m(:n,:n)=U
    m(:,n+1)=y
    call solve_upper_diagonal(m,x)
    write(*,'(f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4)') x
    print *

    end subroutine doolittle_decomposition
```

Result

```
桌面 — bash — 91×50
==============================
Doolittle Decomposition Method
==============================

Exchange Column Pivot Element

 31.0000 -13.0000   0.0000   0.0000   0.0000 -10.0000   0.0000   0.0000   0.0000 -15.0000
-13.0000  35.0000  -9.0000   0.0000 -11.0000   0.0000   0.0000   0.0000   0.0000  27.0000
  0.0000  -9.0000  31.0000 -10.0000   0.0000   0.0000   0.0000   0.0000   0.0000 -23.0000
  0.0000   0.0000 -10.0000  79.0000 -30.0000   0.0000   0.0000   0.0000  -9.0000   0.0000
  0.0000   0.0000   0.0000 -30.0000  57.0000  -7.0000   0.0000  -5.0000   0.0000 -20.0000
  0.0000   0.0000   0.0000   0.0000  -7.0000  47.0000 -30.0000   0.0000   0.0000  12.0000
  0.0000   0.0000   0.0000   0.0000   0.0000 -30.0000  41.0000   0.0000   0.0000  -7.0000
  0.0000   0.0000   0.0000   0.0000  -5.0000   0.0000   0.0000  27.0000  -2.0000   7.0000
  0.0000   0.0000   0.0000  -9.0000   0.0000   0.0000   0.0000  -2.0000  29.0000  10.0000

 LU Decomposition

* lower triangular matrix L

  1.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000
 -0.4194   1.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000
  0.0000  -0.3046   1.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000
  0.0000   0.0000  -0.3539   1.0000   0.0000   0.0000   0.0000   0.0000   0.0000
  0.0000   0.0000   0.0000  -0.3976   1.0000   0.0000   0.0000   0.0000   0.0000
  0.0000   0.0000   0.0000   0.0000  -0.1569   1.0000   0.0000   0.0000   0.0000
  0.0000   0.0000   0.0000   0.0000   0.0000  -0.6540   1.0000   0.0000   0.0000
  0.0000   0.0000   0.0000   0.0000  -0.1121  -0.0175  -0.0246   1.0000   0.0000
  0.0000   0.0000   0.0000  -0.1193  -0.0834  -0.0142  -0.0200  -0.0923   1.0000

* upper triangular matrix U

 31.0000 -13.0000   0.0000   0.0000   0.0000 -10.0000   0.0000   0.0000   0.0000
  0.0000  29.5484  -9.0000   0.0000 -11.0000  -4.1935   0.0000   0.0000   0.0000
  0.0000   0.0000  28.2587 -10.0000  -3.3504  -1.2773   0.0000   0.0000   0.0000
  0.0000   0.0000   0.0000  75.4613 -31.1856  -0.4520   0.0000   0.0000  -9.0000
  0.0000   0.0000   0.0000   0.0000  44.6020  -7.1797   0.0000  -5.0000  -3.5780
  0.0000   0.0000   0.0000   0.0000   0.0000  45.8732 -30.0000  -0.7847  -0.5615
  0.0000   0.0000   0.0000   0.0000   0.0000   0.0000  21.3807  -0.5132  -0.3672
  0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000  26.4131  -2.4200
  0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000  27.3895

Solve Ly=b, get y

-15.0000  20.7097 -16.6921  -5.9069 -22.3483   8.4926  -1.4461   4.6081   7.9492

Solve Ux=y, get x

 -0.2892   0.3454  -0.7128  -0.2206  -0.4304   0.1543  -0.0578   0.2011   0.2902
```

Gauss-Seidel Iteration Method

流程图


Start
Input matrix A and vector b
Set initial x0
xi(k+1)=-1/aii(aij*xj(k+1)-bi)
G/(1-G) * ||x(k+1)-x(k)|| < e
no
yes
Output X
End

Code

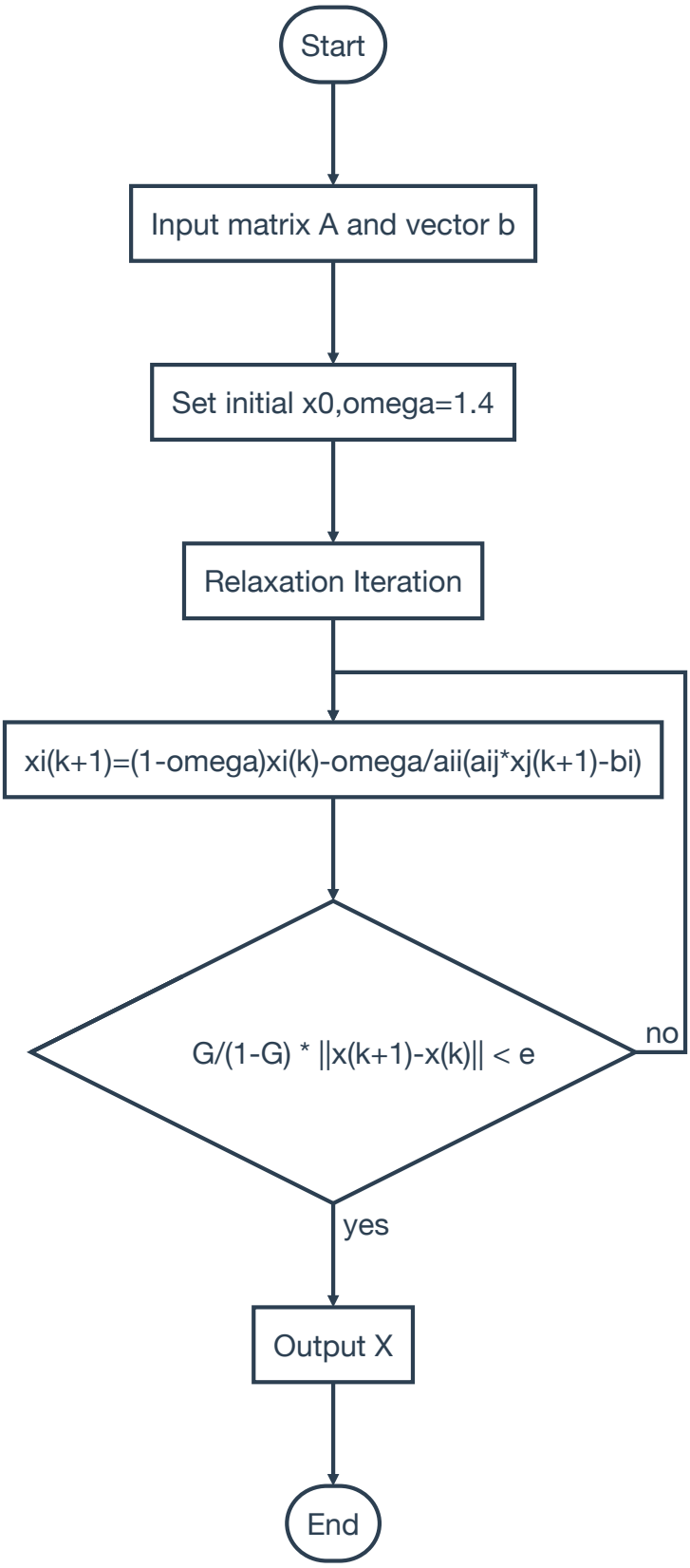
```fortran
! ==============================
! Gauss-Seidel Iteration Method
! when omiga = 1
subroutine gauss_seidel(matrix)
implicit none
real,dimension(9,10) :: m,matrix
real,dimension(9) :: x=0
m=matrix

print *,'==============================='
print *,'Gauss-Seidel Iteration Method'
print *,'==============================='
print *
call relaxation(m,x,1.0)
print *,'The result:'
print *
write(*,'(f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4)') x
print *

end subroutine gauss_seidel
```

其中调用了`relaxation`方法，只是$\omega = 1$。`relaxation`方法如下：

```fortran
! ==================================
! Relaxation Iteration Method
subroutine relaxation(m,x,omiga)
implicit none
real,dimension(9,10) :: m
real,dimension(9) :: x,x_
real :: omiga,e
integer :: n,i
n=size(m(:,1))
e=0.01
print *,'set initial x0 = '
write(*,'(f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4)') x
print *
do
    x_=x
    do i=1,n
        x(i)=x(i)+omiga*(m(i,n+1)-sum(m(i,:n)*x))/m(i,i)
    end do
    if (sum(abs(x-x_)) <= e) exit
end do

end subroutine relaxation
```

Over-relaxation Iteration Method

流程图



Code

```fortran
! ================================
! Over-relaxation Iteration Method
subroutine over_relaxation(matrix)
implicit none
real,dimension(9,10) :: m,matrix
real,dimension(9) :: x=0
real :: omiga=1.40
m=matrix

print *,'================================'
print *,'Over-relaxation Iteration Method'
print *,'================================'
print *
write(*,'(a,f7.4)') ' set omiga =',omiga
print *
call relaxation(m,x,omiga)
print *,'The result:'
print *
write(*,'(f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4,f9.4)') x
print *

end subroutine over_relaxation
```

同理，调用`relaxation`方法，设置$\omega = 1.4$。

Result

桌面 — bash — 91×26

```
==============================
Gauss-Seidel Iteration Method
==============================

set initial x0 =
  0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000

The result:

 -0.2874   0.3469  -0.7120  -0.2201  -0.4299   0.1552  -0.0572   0.2012   0.2904

================================
Over-relaxation Iteration Method
================================

set omiga = 1.4000

set initial x0 =
  0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000

The result:

 -0.2889   0.3453  -0.7132  -0.2208  -0.4305   0.1543  -0.0579   0.2009   0.2901

Eulars-MacBook-Pro:Desktop eular$
```