Aim : To solve the equation using the Gauss Jordan Method and also illustrate the Kirchhoff's Laws.

Theory:

Gauss Jordan Method :

It is algebraic method used to find the inverse of the matrix by converting into diagonal matrix by using the elementary row operations.

Algorithm :

This can be done by using two loop. First loop go from one to one less to the order of the matrix and the second loop go from the iterator of first loop to the order of the matrix and at every point this operation is applied.

$$R_j = R_j - \frac{a_{ji}}{a_{ii}} * R_i$$

Here, I and j are the iterator of the first and second loop respectively.

And the repeat the same process but this time from bottom to up to make the diagonal matrix.

Program in FORTRAN95:

```
program guass_elmination
   implicit none     ! declaring the variables
   real , dimension(:,:),allocatable :: matrix
   real , dimension(:) , allocatable :: variable
   integer :: order , i , j , k , pivot_status
   real :: determinant , sol , mul
   print *, "Enter the number of the variable :: "
   read *, order       ! getting the number of the variable and equation and validating it
   if(order<=0) then
      stop "Invalid number of the variable"
   endif
   allocate(matrix(order,order+1),variable(order))    ! allocate the matrix and the variable array
   ! the variable array use to track the flip column operation and find which variable is where
   ! defining the variable array
   do i = 1,order
      variable(i) = i
   enddo     ! getting the equation the form of the matrix
   print *, "Enter the equation in the form of the matrix :: "
   do i = 1, order
      do j = 1,order+1
         if (j > order) then
            print *, "Enter the constant term in ",i,"th equation :: "
            read *, matrix(i,order+1)
         else
            print *, "Enter ",i,j,"th element :: "
            read *, matrix(i,j)
         endif
      enddo
   enddo
   print *, "Enter the status of the pivot for no pivot chosse 0,for half choose 1, for full choose 2"
   read *, pivot_status
   if(pivot_status < 0 .or. pivot_status > 2) then
```

```
      stop "Invalid pivot option "
   endif    ! making the upper triangular matrix from the given matrix by row operation
   do i = 1,order-1
1     if (matrix(i,i) == 0) then
         do k = i+1,order ! check that the pivot element is zero or not
            if (matrix(k,i) .ne. 0) then
               call flip(k,i) ! changing the pivot element by flipping the row
               goto 1
            endif
         enddo
         if(matrix(i,i) == 0) stop "For these equation a unique solution is not exists"
      endif
      call pivot(i) ! doing pivoting in the matrix with the status user define
      do j = i+1,order
         mul = matrix(j,i)/matrix(i,i)
         do k = i+1,order+1
            matrix(j,k) = matrix(j,k) - (mul)*matrix(i,k)
         enddo
      enddo
   enddo   ! find the determinant of the upper triangular matrix and check it is not equal to zero
   determinant = 1
   do i = 1,order
      determinant = determinant*matrix(i,i)
   enddo
   if (determinant == 0) then
      stop "No Solution is exist for these equation "
   endif      ! making the upper non diagonal element zero
   do i = order,2,-1
      do j = i-1,1,-1
         mul = matrix(j,i)/matrix(i,i)
         matrix(j,order+1) = matrix(j,order+1) - mul*matrix(i,order+1)
      enddo
   enddo
   print *, "The values of the variable are :: "
   do i = 1,order   ! finding the match for the variable and solution of the equations
      print *, "Value of ",i,"th element is :: "
      do j = 1,order
         if (i == variable(j)) then
            print *, matrix(j,order+1)/matrix(j,j)
         endif
      enddo
   enddo
   deallocate(variable,matrix) ! clearing the heap memory by deallocating the all array stored
   stop
   contains   ! subroutine use for the operation
      subroutine flip(row1,row2) ! this subroutine use to flip the rows
         integer ,intent(in)::row1,row2
         real :: temp_element
         integer :: i
         do i = 1,order+1
            temp_element = matrix(row1,i)
            matrix(row1,i) = matrix(row2,i)
```

```
            matrix(row2,i) = temp_element
        enddo
        return
    end subroutine flip
    subroutine flip_col(col1,col2) ! this subroutine use to flip the column
        integer , intent(in) :: col1,col2
        real :: temp_element
        integer :: i
        do i = 1,order
            temp_element = matrix(i,col1)
            matrix(i,col1) = matrix(i,col2)
            matrix(i,col2) = temp_element
        enddo
        temp_element = variable(col1) ! this is also flip the corresponding variables in the variable array
        variable(col1) = variable(col2)
        variable(col2) = temp_element
        return
    end subroutine flip_col
    subroutine pivot(row) ! this subroutine is use for to do pivoting in the matrix per user define
pivoting status (as variable  pivot_status)
        integer , intent(in) :: row
        real :: pivot_value
        integer :: row_max_pivot , i , col_max_pivot , j
        if (pivot_status == 0)  return ! Return  just end the function
        else if (pivot_status == 1) then ! for half pivoting
            pivot_value = abs(matrix(row,row))
            do i = row,order
                if (abs(matrix(i,row)) > pivot_value) then
                    row_max_pivot = i
                    pivot_value = abs(matrix(i,row))
                endif
            enddo
            call flip(row,row_max_pivot) ! only flip the row which have maximum pivot element
            return
        else if (pivot_status == 2) then ! for do full pivoting in the matrix
            pivot_value = abs(matrix(row,row))
            do i = row,order
                do j = row,order
                    if (abs(pivot_value) < matrix(i,j)) then
                        pivot_value = abs(matrix(i,j))
                        row_max_pivot = i
                        col_max_pivot = j
                    endif
                enddo
            enddo
            call flip(row,row_max_pivot) ! here we check the element in the whole remaining matrix
            call flip_col(row,col_max_pivot)
            return
        endif
        return
    end subroutine pivot
end program
```

Flow Chart :

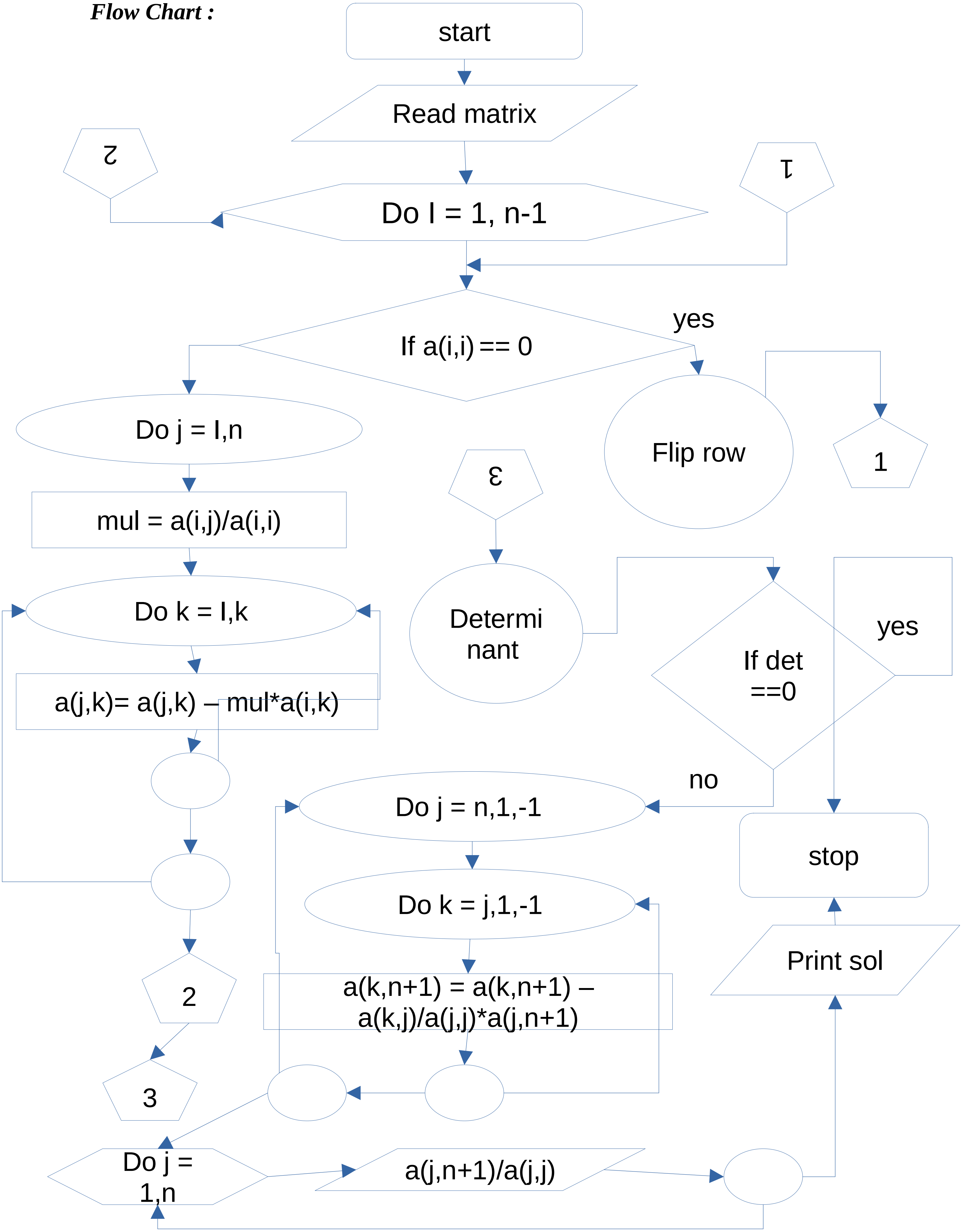
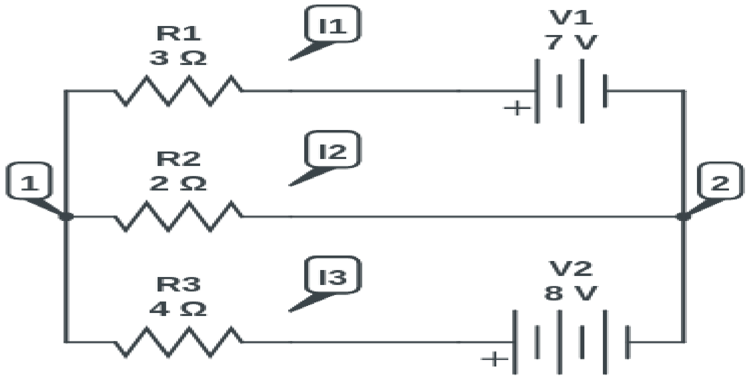

start
Read matrix
2
1
Do I = 1, n-1
If a(i,i) == 0
yes
Flip row
1
Do j = I,n
3
mul = a(i,j)/a(i,i)
Determi
nant
Do k = I,k
If det
==0
yes
a(j,k)= a(j,k) – mul*a(i,k)
no
Do j = n,1,-1
stop
Do k = j,1,-1
Print sol
2
a(k,n+1) = a(k,n+1) –
a(k,j)/a(j,j)*a(j,n+1)
3
Do j =
1,n
a(j,n+1)/a(j,j)

Illustration of Kirchhoff's Law :

Circuit :



Equations for the current value using Kirchhoff's Law:

$$I_1 - I_2 + I_3 = 0 \qquad R_1 I_1 + R_2 I_2 = V_1 \qquad R_2 I_2 + R_3 I_3 = V_2$$

Output of the program :

```
 Enter the number of the variable ::
3
 Enter the equation in the form of the matrix ::
 Enter           1           1 th element ::
1
 Enter           1           2 th element ::
-1
 Enter           1           3 th element ::
1
 Enter the constant term in           1 th equation ::
0
 Enter           2           1 th element ::
3
 Enter           2           2 th element ::
2
 Enter           2           3 th element ::
0
 Enter the constant term in           2 th equation ::
7
 Enter           3           1 th element ::
0
 Enter           3           2 th element ::
2
 Enter           3           3 th element ::
4
 Enter the constant term in           3 th equation ::
8
 Enter the status of the pivot for no pivot choose 0,for half choose 1, for full choose 2
2
 The values of the variable are ::
 Value of           1 th element is ::
   1.00000000
 Value of           2 th element is ::
   2.00000000
 Value of           3 th element is ::
   1.00000000
```