```
!Apéndice al Capítulo 2.
!Programa para calcular la funcion de autocorrelacion para un sitio
!superficial acoplado debilmente a una cadena tight-bindig.
program cadenas

use msflib
use msimsl
use portlib

implicit none

!Defino todas las variables

!Variable de control de salida
logical salir
!variables para bucles y cuentas intermedias
integer n,n2,p ,k
real(8) rtemp, rtemp2
complex(8) comp
character(60) mensaje,ctemp
!unidad imaginaria
complex(8), parameter :: I = (0,1)
!Nombre del archivo a guardar, directorio a usar
character(60) archivodatos ,dir
!momento de inicio y de final de la cuenta
integer time1, time2
!cantidad de pasos para la evolucion
integer, parameter :: steps=1000
!tiempo, Pi
real(8) t, Pi
!Funcion de autocorrelacion, array de tiempos
real (8),allocatable :: P11(:),P11p(:),rtiempo(:)
!Cantidad de sitios de la cadena (incluyendo el superficial)
integer Ntotal
!Energia Eo de la cadena, hoping Vs superficial
!hoping V de la cadena, Energia Es superficial
real(8) Eo, Vs,V,Es
!tiempo final, tiempo inicial
real (8) tiempo, tiempoini

!Valores iniciales de algunas variables
data Ntotal /100/
data Eo /0/, V /1.0/, Vs /0.1/ , Vpar /4.0/, W /4/
data tiempo /120/, tiempoini /0.0/
!Directorio donde se grabaran los datos
data dir /'C:\datos\'/

!Defino Vectores y matrices para el calculo con memoria dinamica
complex(8),allocatable :: autovals(:)
real(8),allocatable :: H(:,:)
complex(8),allocatable :: ket(:),vector(:),bra(:)
complex(8),allocatable :: invbase(:,:)

salir = .true.
dir = adjustl(dir)
!Valor de Pi
Pi=4*atan(1.0)

principalout: do while (salir)
       write(6,*) 'Ingrese nombre del archivo de salida: ( 0 para salir)'
       read (5,*) archivodatos
       if (archivodatos.eq.'0') then !me fijo si N=0, si es asi pongo salir a .f. y hago un cycle
              salir = .false.
              cycle
       endif
       write(6,*) 'Ingrese largo de la cadena'
       read (5,*) Ntotal
       write(6,*) 'Ingrese tiempo inicial'
       read (5,*) tiempoini
       write(6,*) 'Ingrese tiempo final'
       read (5,*) tiempo
       write(6,*) 'Ingrese hoping V'
       read (5,*) V
       write(6,*) 'Ingrese hoping superficial Vs'
       read (5,*) Vs
       write(6,*) 'Ingrese Energia superficial Es'
       read (5,*) Es

       write(6,*) 'Calculando...'
```

```
!Guardo el archivo de informacion
open (4, FILE = trim(dir)//trim(archivodatos)//'.log',ACTION='WRITE', STATUS = 'UNKNOWN')
       write (4,*) 'Cadena con sitio superficial debilmente acoplado'
       write (4,*) 'Formato'
       write (4,*) 't        P11(t)'
       write (4,*) 'Largo de la cadena     : ', Ntotal
       write (4,*) 'Hoping V               : ', V
       write (4,*) 'Hoping Vs              : ', Vs
       write (4,*) 'Energia Eo             : ', Eo
       write (4,*) 'Energia Es             : ', Es
       write (4,*) 'Tiempo inicial         : ', tiempoini
       write (4,*) 'Tiempo final           : ', tiempo
       write (4,*) 'Pasos                  : ', steps
       call time(mensaje)
       call date(ctemp)
       write (4,*) 'Fecha y Hora           : ', trim(ctemp),' ',trim(mensaje)
close (4)

!Asigno la memoria para las variables
       allocate (autovals(Ntotal))
       allocate (H(Ntotal,Ntotal))
       allocate (ket(Ntotal),vector(Ntotal))
       allocate (invbase(Ntotal,Ntotal))
       allocate (rtiempo(steps+1),P11(steps+1))
       allocate (bra(Ntotal))

print *, 'Memoria asignada, diagonalizando'

!Cuento el tiempo de comienzo
time1 = time()

! Inicializo el Hamiltoniano
H = H*0

! lleno el Hamiltoniano de valores
do n=1,Ntotal-1
       H(n+1,n)=V
       H(n,n+1)=V
       H(n,n)=Eo
enddo
H(Ntotal,Ntotal)=Eo
H(1,1)=Es
H(1,2)=Vs
H(2,1)=Vs

! calculo autovectores y autovalores
call devcrg (Ntotal, H, Ntotal, autovals, invbase,Ntotal)

print *, 'Diagonalizacion terminada'

!_____________________________________________________________
! COMIENZA EL CALCULO DE LA EVOLUCION
!PARA EL ESTADO INICIAL

!Pongo el estado inicial en el sitio superficial
do n = 1,Ntotal
       ket(n) = invbase(1,n)
enddo

print *, 'Comienza evolucion'

principal: do n = 0,steps

       !Calculo el tiempo en dimensiones de h/V
       t = ((n*(tiempo-tiempoini))/steps + tiempoini)/V ! (hbarra=1)

       !Hago la evolucion en |vector>
       interno: do n2 = 1, Ntotal
              vector(n2) = ket(n2) * cdexp(-I* t * autovals(n2))
       enddo interno
       !multiplico <bra|vector>
       comp = dot_product (ket, vector)
       !Calculo la funcion de autocorrelacion P11=|<bra|vector>|^2
       P11(n+1) = comp * dconjg (comp)

       !Guardo el tiempo en el array
       rtiempo(n+1) = t

end do principal
! TERMINA LA EVOLUCION
```

```
!_____________________________________________________________

!Grabo los datos
open (4, FILE = trim(dir)//trim(archivodatos)//'.dat',ACTION='WRITE', STATUS = 'UNKNOWN')
do n=0,steps
       write (4,*) rtiempo(n+1),'    ',P11(n+1)
enddo
close (4)

!Libero la memoria
deallocate (autovals)
deallocate (H)
deallocate (ket,vector)
deallocate (invbase)
deallocate (rtiempo,P11)
deallocate (bra)

print *, 'Evolucion terminada'

!Calculo y muestro el tiempo utilizado
time2 = time()

time1 = time2 - time1

print *, 'Tiempo total [s]' ,time1

enddo  principalout

end program cadenas
```

```
!Apéndice al Capítulo 3.
~
!Programa para calcular promedios de Pii sobre i (sitios de una capa)
!y diferentes corridas de desorden.
!Se utilizó para observar comportamiento difusivo

program difusion

use msflib
use msimsl
use portlib

implicit none

!Defino todas las variables

!Variable de control de salida
logical salir

!variables para bucles y cuentas intermedias
integer n,n2,n3,m,mini,x,y,z,p,pw ,pt
real(8) t, Pi, rtemp, rtemp2
complex(8) comp
character(60) mensaje,ctemp
character(60) archivodatos ,dir
real(4) azar

!unidad imaginaria
complex(8), parameter :: I = (0,1)
!cantidad de pasos para la evolucion
integer, parameter :: steps=500
!Promedios en sitios y en desorden
integer promedios, promW
!momento de inicio y de final de la cuenta
integer time1, time2
!Sitios en una capa, sitios a lo largo del sistema
!cantidad total de sitios, posicion inicial en la capa, posicion inicial en el largo
integer Nxy,Nz,Ntotal, XYini, Zini
!Energia de la cadena, hoping en la capa, hoping entre capas, Desorden
real(8) Eo, Vxy,Vz, W
!Tiempo final e inicial para la evolucion
real (8) tiempo, tiempoini

real (8),allocatable :: re(:,:),Poo(:,:)

!Valores iniciales de algunas variables
data Nxy /12/, Nz /100/
data XYini /1/, Zini /1/
data Eo /0.0/, Vxy /1.0/, Vz /1.0/, W /1/
data tiempo /60/, tiempoini /0/
!Directorio donde se almacenan los resultados
data dir /'d:\Fernando\fortran Projects\datos\'/

!Defino Vectores y matrices para el calculo con memoria dinamica
complex(8),allocatable :: autovals(:)
real(8),allocatable :: H(:,:)
complex(8),allocatable :: ket(:),vector(:),bra(:)
complex(8),allocatable :: ket0(:),vector0(:)
complex(8),allocatable :: invbase(:,:)

salir = .true.
dir = adjustl(dir)

principalout: do while (salir)
       write(6,*) 'Ingrese nombre del archivo de salida:'
       read (5,*) archivodatos
       if (archivodatos.eq.'0') then !me fijo si N=0, si es asi pongo salir a .f. y hago un cycle
              salir = .false.
              cycle
       endif
       !Se ingresan los datos
       write(6,*) 'Nxy Nz'
       read (5,*) Nxy
       read (5,*) Nz
       Ntotal = Nxy*Nz
       write(6,*) 'Tiempo inicial'
       read (5,*) tiempoini
       write(6,*) 'Tiempo final'
       read (5,*) tiempo
       write(6,*) 'Energia de la cadena Eo'
```

```
        read (5,*) Eo
        write(6,*) 'Hoping entre capas Vz'
        read (5,*) Vz
        write(6,*) 'Hoping en la capa Vxy'
        read (5,*) Vxy
        write(6,*) 'Desorden'
        read (5,*) W
        write(6,*) 'Posicion inicial en la capa'
        read (5,*) XYini
        write(6,*) 'Posicion inicial en el largo'
        read (5,*) Zini
        write(6,*) 'Promedios en sitios'
        read (5,*) promedios
        write(6,*) 'Promedios en desordenes'
        read (5,*) PromW

!Grabo el archivo de informacion de la corrida
open (4, FILE = trim(dir)//trim(archivodatos)//'.log',ACTION='WRITE', STATUS = 'UNKNOWN')
        write (4,*) 'Barras para calcular difusion'
        write (4,*) 'Formato'
        write (4,*) 't        Suma(capa)<<Pii(t)>i>w    <Poo(t)>iw'
        write (4,*) 'Dimensiones (xy,z)     : ', Nxy,' ', Nz
        write (4,*) 'Hoping Vxy              : ', Vxy
        write (4,*) 'Hoping Vz              : ', Vz
        write (4,*) 'Energia Eo             : ', Eo
        write (4,*) 'Desorden               :  ',W
        write (4,*) 'Tiempo inicial         : ', tiempoini
        write (4,*) 'Tiempo final           : ', tiempo
        write (4,*) 'Pasos                  : ', steps
        write (4,*) 'Promedios en sitios    : ', promedios
        write (4,*) 'Promedios en desorden  : ', promW
        call time(mensaje)
        call date(ctemp)
        write (4,*) 'Fecha y Hora           : ', trim(ctemp),' ',trim(mensaje)
close (4)

        write(6,*) 'Calculando...'

        !Asigno la memoria a los arrays
        allocate (autovals(Ntotal))
        allocate (H(Ntotal,Ntotal))
        allocate (ket(Ntotal),vector(Ntotal), bra(Ntotal))
        allocate (ket0(Ntotal),vector0(Ntotal))
        allocate (invbase(Ntotal,Ntotal))
        allocate (re(steps+1,promedios*promW),Poo(steps+1,promedios*promW))

print *, 'Memoria asignada, Calculando'

time1 = time()

!Valor de PI
Pi=4*atan(1.0)

!Uso una semilla aleatoria con el tiempo
call seed(rnd$timeseed)

! Este es el bucle que se encarga de calcular
! promedios para diferentes corridas de desorden
promgrande: do pw=0,promW-1

        print *, (100*pw)/promW,' % realizado', (time()-time1),' segundos de calculo'

        ! Inicializo algunas matrices y vectores a cero
        H = H*0

!-------------------------------------
! En esta seccion se inserta alguno de
! los Hamiltonianos de capa que se
! muestran despues del programa
!-------------------------------------

        ! Aqui se llena la parte del Hamiltoniano
        ! entre capas
        do z=1,Nz
                do x=1,Nxy
                        m= (z-1)*Nxy + x
                        call random(azar)
                        H(m,m) = (azar-0.5)*W
                        if ((m+Nxy).gt.Ntotal) then
                                H(m,m+Nxy-Ntotal) = Vz
```

```
                        H(m+Nxy-Ntotal,m) = Vz
                    else
                        H(m,m+Nxy) = Vz
                        H(m+Nxy,m) = Vz
                    endif
                enddo
        enddo


        ! calculo autovectoresy autovalores
        call devcrg (Ntotal, H, Ntotal, autovals, invbase,Ntotal)

        print *, 'Diagonalizacion terminada', (time()-time1), ' s'

! _____________________________________________________
! COMIENZA EL CALCULO DE LA EVOLUCION
!PARA EL ESTADO INICIAL

! Este es el bucle que se encarga de calcular promedios
! sobre los sitios del sistema
        promediacion: do pt = 1,promedios

                p=promedios*pw+pt

                ! Calculo la capa donde se va a poner el paquete inicial
                mini = (pt-1)*Nxy + XYini

                ! El paquete inicial es una combinacion cualquiera de
                ! los sitios de una capa, este es el que se usa para
                ! calcular la funcion de autocorrelacion a la capa
                ket=ket*0
                do n2= 0,Nxy-1
                        call random(azar)
                        do n = 1,Ntotal
                                ket(n) = 2*(azar-0.5)*invbase(mini+n2,n)+ket(n)
                        enddo
                enddo
                rtemp = dot_product (ket, ket)
                ket=ket/rtemp**0.5

                ! Este es un paquete inicial para calcular la funcion
                ! de autocorrelacion del sitio
                do n = 1,Ntotal
                        ket0(n) = invbase(mini,n)
                enddo

                principal: do n = 0,steps

                        !Pongo el tiempo en dimensiones de h/V
                        t = ((n*(tiempo-tiempoini))/steps + tiempoini)! * (hbarra/hoping)  t          =
temp(n+1)

                        !Hago la evolucion en |v> y |v'>
                        interno: do n2 = 1, Ntotal
                                vector(n2) = ket(n2) * cdexp(-I* t * autovals(n2))
                                vector0(n2) = ket0(n2) * cdexp(-I* t * autovals(n2))
                        enddo interno

                        !Calculo la autocorrelacion al sitio
                        comp = dot_product (ket0, vector0)
                        Poo(n+1,p)=comp * dconjg (comp)
                        rtemp=0

                        ! Calculo la autocorrelacion a la capa
                        do n2 = 0, Nxy-1
                                do n3 = 1,Ntotal
                                        bra(n3) = invbase(mini+n2,n3)
                                enddo
                                comp = dot_product (bra, vector)
                                rtemp = rtemp + comp * dconjg (comp)
                        enddo
                        re(n+1,p) = rtemp

                end do  principal

! TERMINA LA EVOLUCION
!_____________________________________________________

        enddo promediacion
enddo promgrande
```

```
! Calculo los promedios y grabo los resultados
open (4, FILE = trim(dir)//trim(archivodatos)//'.dat',ACTION='WRITE', STATUS = 'UNKNOWN')
print *, 'calculando promedios', (time()-time1), ' s'
do n=0,steps
       rtemp=0
       rtemp2=0
       do n2=1,promedios*promW
              rtemp = re(n+1,n2)+rtemp
              rtemp2= Poo(n+1,n2) + rtemp2
       enddo
       rtemp=rtemp/(promedios*promW)
       rtemp2=rtemp2/(promedios*promW)
       if (n.gt.steps) then
              cycle
       else
              t = ((n*(tiempo-tiempoini))/steps + tiempoini)
              write (4,*) t,'       ',rtemp ,' ',rtemp2
       endif
enddo
close (4)

       !Devuelvo la memoria utilizada
       deallocate (H)
       deallocate (re,Poo)
       deallocate (autovals,invbase)
       deallocate (ket,vector,bra)
       deallocate (ket,vector)

print *, 'Evolucion terminada'

! Calculo el tiempo utilizado para toda la cuenta
time2 = time()

time1 = time2 - time1

print *, 'Tiempo total [s]' ,time1

salir = .false.

enddo  principalout

end program difusion

!----------------------------------------
! A continuacion el codigo que se encarga
! de llenar los diferentes Hamiltonianos
!----------------------------------------

! Este Hamiltoniano de capa corresponde a el sistema estrella
do z=1,Nz
       m= (z-1)*Nxy + 1
       do n=m, m+(Nxy-1)
              do n2=m,m+(Nxy-1)
                     H(n,n2)=Vxy/(Nxy)**.5
              enddo
       enddo
enddo

! Este Hamiltoniano de capa corresponde a el sistema cinta
do z=1,Nz
       m= (z-1)*Nxy + 1
       do n=m, m+(Nxy-1)-1
              H(n,n+1)=Vxy
              H(n+1,n)=Vxy
       enddo
enddo

! Este Hamiltoniano de capa corresponde a el sistema cilindro
do z=1,Nz
       m= (z-1)*Nxy + 1
       do n=m, m+(Nxy-1)-1
              H(n,n+1)=Vxy
              H(n+1,n)=Vxy
       enddo
       H(1,Nxy)=Vxy
       H(Nxy,1)=Vxy
enddo
```

```
C       Apéndice al capitulo 3
~
C-----------------------------------------------------------------------
C       Calcula los exponentes caracteristicos de Lyapunov en una cinta
C       Con el menor de ellos se calcula la longitud de localizacion
C-----------------------------------------------------------------------

        program Lyapunov

        IMPLICIT REAL*4 (Z)
        IMPLICIT REAL*8 (E-H,Q-Y)
         IMPLICIT COMPLEX*16 (A-D,O,P)
        IMPLICIT INTEGER (i-n)

        PARAMETER (rPI=3.1415926535897932D0)
        PARAMETER (N1=5) !Numero de sitios de la capa (canales)
        PARAMETER (N2=2*N1)

         DIMENSION A(N1,N1),B(N1,N1),C(N1,N1),E(N1),AN(N2),zAPROX(N1)
        DIMENSION  SYY(N1),SXY(N1),SY(N1),zLYAP(N1),rLOGN(N1),zERR(N1)
        CHARACTER(60) dir,file

         N=N1

        dir='C:\datos\'

        WRITE(6,*) 'Nombre del archivo'
        READ(5,*) file

C       Aqui se ingresan la energia de Fermi minima y maxima
C       La canidad de pasos , el desorden, la energia a investigar
C       y el largo del sistema
        WRITE(6,*)'min max Ef?,N steps? , Desorden?, Energia?, Largo?'
       READ(5,*)Fmin,Fmax,Nfljos, W, Ef, IMAX

        Finc=(Fmax-Fmin)/Nfljos

        OPEN(UNIT=8, STATUS='UNKNOWN', FILE=trim(dir)//trim(file)//'.dat')
        OPEN(UNIT=9, STATUS='UNKNOWN', FILE=trim(dir)//trim(file)//'.log')

        WRITE(9,*) 'Ancho de la barra ', N
        WRITE(9,*) 'Desorden Min ', Fmin
        WRITE(9,*) 'Desorden Max ', Fmax
        WRITE(9,*) 'Steps ', Nfljos
        WRITE(9,*) 'Energia ', W
        WRITE(9,*) 'Largo Maximo ', IMAX

         Vx=1./(N1)**.5

         cIM=(0.D0,1.D0)

C       Bucle para variar la energia de Fermi
        DO 1000 Ef=Fmin,Fmax, Finc

         SEM1=.23821
         ISEM2=91482
        print *, (100*(Ef-Fmin))/(Fmax-Fmin),' % realizado'
         DO 102 I=1,N
         DO 101 J=1,I
         A(J,I)=(0.D0,0.D0)
         A(I,J)=A(J,I)
         B(J,I)=(0.D0,0.D0)
101     B(I,J)=B(J,I)

        B(I,I)=(1.D0,0.D0)

         zLYAP(I)=0.
C              I-esimo exponente carateristico de Lyapunov

         rLOGN(I)=0.
C              !logaritmo de la norma del subespacio invariante
C                  !minimal
        SYY(I)=0.
        SXY(I)=0.
        SY(I)=0.
102     continue
         ILAR=0
        SX=0.
        SXX=0.
        Nptos=0
```

```
C             !longitud del cilindro

55     CONTINUE
C*****************COMIENZA EL GRAN LAZO de multiplicaciones

        DO 44 IPAR=1,15
C              ! se efectuan IPAR multiplicaciones

        DO 20 I=1,N
C              !W es el desorden, ENI es la energia de sitio
        ENI=W*(RANDU(SEM1)-.5D0)
20      E(I)=(Ef-ENI)

        DO 1 I=1,N
        DO 2 J=1,N
         C(I,J)=E(I)*B(I,J)-A(I,J)
              DO 3 K=1,I-1
3                    C(I,J)=C(I,J)-B(K,J)*Vx
              DO 4 K=I+1,N
4                    C(I,J)=C(I,J)-B(K,J)*Vx

2       CONTINUE
1       CONTINUE

        DO 104 I=1,N
        DO 103 J=1,I
        A(J,I)=B(J,I)
103     A(I,J)=B(I,J)
104     CONTINUE

        DO 106 I=1,N
        DO 105 J=1,I
        B(J,I)=C(J,I)
105     B(I,J)=C(I,J)
106     CONTINUE

        INT=IPAR
        IF(cdabs(A(1,1)).GT.10)GO TO 12

44     CONTINUE
C     termino el lazo de multiplicaciones minimas
C*****************************************************************
C        se acumula el largo del sistema considerado
12      ILAR=ILAR+INT
       rILAR=ILAR

C-------------------------------------------------------------------------C
C                                                                         C
C                 PROCESO DE ORTONORMALIZACION                            C
C                                                                         C
C-------------------------------------------------------------------------C
        QNOR=0.D0

        DO 108 I=1,N
108     QNOR=QNOR+cdabs(B(I,1))**2+cdabs(A(I,1))**2

        QNOR=dsqrt(QNOR)
       rLOGN(N1)=rLOGN(N1)+dlog(QNOR)

        DO 110 I=1,N
        B(I,1)=B(I,1)/QNOR
110     A(I,1)=A(I,1)/QNOR

        DO 66 I=2,N

        DO 112 K=1,N
        AN(K)=B(K,I)
       KmN=K+N
112     AN(KmN)=A(K,I)

C          !Restamos a la primera estimacion su componente sobrep"
C                  !los N-I anteriores
        DO 118 J=1,I-1
C       Calculo de la proyeccion de AN sobre el j-esimo vector que esta
C       en la columna j de B y A
        CC=(0.D0,0.D0)
        DO 114 L=1,N
114     CC=CC + dconjg(B(L,J)) * B(L,I) + dconjg(A(L,J)) * A(L,I)
```

```
C              ahora restamos la proyeccion
        DO 116 K=1,N
        AN(K)=AN(K)-CC*B(K,J)
       KmN=K+N
116    AN(KmN)=AN(KmN)-CC*A(K,J)

118    CONTINUE

        QNOR=0.D0
C              Calculo de la norma de AN
        DO 120 K=1,2*N
120    QNOR=QNOR+dconjg(AN(K))*AN(K)

        QNOR=dsqrt(QNOR)

       rLOGN(N-I+1)= rLOGN(N-I+1) + dlog(QNOR)

        DO 122 K=1,N
        B(K,I)=AN(K)/QNOR
122    A(K,I)=AN(K+N)/QNOR

66     continue
C********Termina el proceso de ortonormalizacion de los vectores
C        acumula el logaritmo de la norma minima

       if(ILAR.GE.IMAX/2) then
       Nptos=Nptos+1
       SX=SX+rILAR
       SXX=SXX+rILAR**2

       do 77 K=1,N1
       SY(K)=SY(K)+rLOGN(K)
       SYY(K)=SYY(K)+rLOGN(K)**2
77     SXY(K)=SXY(K)+rLOGN(K)*rILAR
       end if

       IF(ILAR.LT.IMAX) GO TO 55

       if(Nptos.GE.2)THEN
       do 78 K=1,N1
       rAUX=Nptos*SXY(K)-SX*SY(K)
       rLYAP=rAUX/(Nptos*SXX-SX**2)
       rCORR=rAUX/dsqrt((Nptos*SXX-SX**2)*(Nptos*SYY(K)-SY(K)**2))
       if(rLYAP.LT.0.)rLYAP=0.
       zLYAP(K)=rLYAP
       zAPROX(K)=rLOGN(K)/rILAR
78     zERR(K)=rCORR
       zFc=Ef

C      Guardo todos los coeficientes de Lyapunov del sistema
       write(8,'(20G22.15)') zFc, (zLYAP(K),K=1,N1)
       ENDIF

1000   CONTINUE
        END

       REAL*8 FUNCTION RANDU(DSEED)
       DOUBLE PRECISION DSEED
       DOUBLE PRECISION D2P31M, D2P31
       DATA  D2P31M/2147483647.D0/,D2P31/2147483649.D0/
       DSEED=DMOD(16807.D0*DSEED,D2P31M)
       RANDU=DSEED/D2P31
       RETURN
       END
```

```
c       Apéndice al capitulo 3
~
c       Programa para calcular el camino libre medio de un Hamiltoniano
~
C        en el sistema cinta
~


~
c  ************************
~
        program LMEDIO
~
        use portlib
~


~
C-------------------------------------------------------------
C       Ef = Energia de Fermi investigada
C       N  = Numero de canales (sitios por capa)
C       Vx = Hoping entre capas
C       Vy = Hoping en la capa
C       cSL , cSR matrices complejas con las renormalizaciones
C              de los alambres
C       E_Hef  = Energia de Fermi menos el Hamiltoniano Efectivo
C       gi             =   Localizador de capa renormalizado
C       cGLR   = Matriz de la Funcion de Green final
C       rmodGo  = Funcion de Green para el sistema ordenado
c*************************************************************
        IMPLICIT REAL*8 (E,F,O-Y)
         IMPLICIT COMPLEX*16 (A-d,G-H)
        IMPLICIT INTEGER (I-N)

        PARAMETER (rPI=3.1415926535897932D0)
        PARAMETER (N=6)
        parameter(N2=N*2)
        PARAMETER (NF=500)
        character(60) filename, dir

         DIMENSION  cSR(N,N), cSL(N,N), Gi(N,N),cSwap(N,N)
     1     ,cVeff(N,N),ctVeff(N,N),rmodG(N)
     2  , cEHeff(N2,N2),cGLR(N2,N2),cSI(N,N),rmodGo(NF,N)
     3  ,cEH(N,N),cG0L(N,N),cG00(N,N),cSk(N),rT(4,4),cGreen(N)

        double precision RANDU
        complex*16 cS

        dir='c:\datos\'

        write(6,*) 'Ingrese nombre del archivo de salida:'
        read (5,*) filename

        WRITE(6,*)'Emin Emax?, DESORDEN?, Largo, Promeds?'
        READ(5,*)Fmin, Fmax, W, Lm1, nPromedios

        L=Lm1+1
        Np=N
        Fc=0

        open (9,FILE=trim(dir)//trim(filename)//'.log',STATUS='UNKNOWN')

        WRITE(9,*) 'Calculo de l medio en una rueda'
        WRITE(9,*) 'Ancho(Canales:M) ',N
        WRITE(9,*) 'Largo de la rueda (r)',Lm1
       WRITE(9,*) 'Flujo', Fc
        WRITE(9,*) 'Desorden ', W
         WRITE(9,*) 'Energia Min', Fmin
        WRITE (9,*) 'Energia Max', Fmax
        WRITE(9,*) 'Numero de energias ',NF
        WRITE(9,*) 'Numero de Promedios ',nPromedios
        WRITE(9,*) 'Salida:'
        WRITE(9,*) '(Efermi+1/M**0.5) <l>medio  <l>suma'

         close(9)

        Vx=1.D0
        Vy=1

         cIM=(0.D0,1.D0)
         cRE=(1.D0,0.D0)
```

```
        Fmin=Fmin-Vy
        Fmax=Fmax-Vy

c-------------------------------------------------------------
c  CALCULO DEL MODULO DE G PARA EL SISTEMA ORDENADO
C-------------------------------------------------------------
        ntime1=time()
        W2=W
        W=0
       DO 1999 nEf=0,NF-1
        Ef=(nEf*(Fmax-Fmin)/NF)+Fmin

        cGreen=cGreen*0
        print *,( (100*nEf)/NF), ' % ',(time()-ntime1),' s'

       DO  I=1,N
         cEH(I,I)=(0.D0,0.D0)

           DO  J=I+1,N
          cEH(I,J)=(0.D0,0.D0)
          cEH(J,I)=(0.D0,0.D0)
          enddo
        enddo
C*********************************************************
C-defino las autoenergias y los hoping efectivos (prcedimiento de decimation)
c        ____________________________
c        inicializo variables del Hamiltoniano efectivo
c        SR=0, SL=0, Veff=Vx*rUNIT

        do i=1,N
         do j=1,N
          cSR(i,j)=(0.d0,0.d0)
          cSL(i,j)=(0.d0,0.d0)
          cVeff(i,j)=(0.d0,0.d0)
         enddo
          cVeff(i,i)=Vx*cRE
        end do
C        -------------------------------
c       comienza eliminacion de (Lm-1) capas internas del sistema

        if (Lm1.ge.1) then
         do it=1,Lm1 !toma cada capa

               cHOP=Vy
               cHOPc=dconjg(cHOP)

               do I=1,N
                      Iu=I-1
                      cEH(I,I)=(Ef)*cRE !+(0.0001*cIM)
                      If(I.GT.1) then
                             cEH(I,Iu)=-cHOP
                             cEH(Iu,I)=-cHOPC
                      end if
               enddo !I

            call cSUM(Gi,1.d0,cEH,-1.d0,cSR,N)
            call cINV(Gi,N)

c           *****SL=SL+Veff(1/(E-Ei-SR))Veff*****
            call cPROD(cSwap,1.d0,cVeff,Gi,N)
            call cTRAN(ctVeff,cVeff,N)
            call cPROD(cSR,1.d0,cSwap,ctVeff,N)
            call cSUM(cSL,1.d0,cSL,1.d0,cSR,N)

c           ******SR=(V(i+1)**2)/(E-Ei-SR)*******
             call cEQUAL(cSR,Vx**2,Gi,N)

c           ******Veff=Veff*1/(E-Ei-SR)*V(i+1)******
            call cEQUAL(cVeff,Vx,cSwap,N)
          end do
         end  if !si el cuadro es 2x2 salta estas operaciones.

c Defino las autoenergias de los alambres externos (en la base k)

        do k=1,N
        cSk(k)=cS( Ef-2.D0*Vy*dcos(k*rPI/Np) , Vx )
        end do

c      defino la matriz de las autoenergias de los alambres en la base de sitio
               do I1=1,N
```

```
                      do I2=1,N
                             ctmp=(0.d0,0.d0)
                             do k=1,N
                                    ctmp=ctmp+(2.D0/Np)*dsin(I1*k*rPI/Np)*dsin(I2*k*rPI/Np)*cSk(k)
                             end do !k
                             cSI(I1,I2)=ctmp
                      end do !I2
                end do !I1
C Defino el Hamiltoniano efectivo escrito en la base de la primera y
C ultima capa del sistema teniendo en cuenta los alambres

c     ___________Comienzo definiendo las contribuciones
C    del sistema y las decimaciones de los alambres
       do i=1,N
           do j=1,N
c            Pone la contribucion de los sitios eliminados
           cEHeff(i,j)=    -cSL(i,j) -cSI(i,j)
           cEHeff(N+i,N+j)=-cSR(i,j) -cSI(i,j)
           cEHeff(i,N+j)=      -cVeff(i,j)
           cEHeff(N+j,i)=  -dconjg(cVeff(i,j))
           end do !j

c     Aagrego las energias diagonales no perturbadas de la capa
          cEHeff(i,i)=cEHeff(i,i)+(Ef)*cRE
          cEHeff(N+i,N+i)=cEHeff(N+i,N+i)+(Ef)*cRE
       enddo !i
c     Agrego los hopings en la capa
       do i=1,N-1
       i2=i+1
                      cEHeff(i,i2)=cEHeff(i,i2)-Vy
                      cEHeff(i2,i)=cEHeff(i2,i)-Vy
                      cEHeff(N+i,N+i2)=cEHeff(N+i,N+i2)-Vy
                      cEHeff(N+i2,N+i)=cEHeff(N+i2,N+i)-Vy
       end do   ! i


       call cEQUAL(cGLR,1.d0,cEHeff,N2)
       call cINV(cGLR,N2)
c Transformo la funcion de Green a la base k
       do k1=1,N
       do k2=1,N
       ct=(0.D0,0.D0)
       c2=(0.D0,0.D0)
         do I1=1,N
         do I2=1,N
       ct=ct+(2.D0/Np)*dsin(I1*k1*rPI/Np)*cGLR(I1,N+I2)
       1                    *dsin(I2*k2*rPI/Np)
       c2=c2+(2.D0/Np)*dsin(I1*k1*rPI/Np)*cGLR(I1,I2)*dsin(I2*k2*rPI/Np)
        end do
        end do
       cG0L(k1,k2)=ct
       cG00(k1,k2)=c2
       end do
       enddo

c***Calculo del modulo de la funcion de Green para
c el sistema ordenado, Go

       Do k1=1,N
      rmodGo(nEf+1,k1)=cdabs(cG0L(k1,k1))
       if (rmodGo(nEf+1,k1).eq.0) rmodGo(nEf+1,k1)=10**(-15)
       end do !k1

1999   Continue

c--------------------------------------------------------------
c  --------------Comienza el calculo del camino libre medio-------

       W=W2
       ntime1=time()

      DO 1000 nEf=0,NF-1
       Ef=(nEf*(Fmax-Fmin)/NF)+Fmin
       cGreen=cGreen*0
       print *,( (100*(Ef-Fmin))/(Fmax-Fmin)), ' % ',(time()-ntime1),' s'
       SEM1=.23821

       do 9999 nprom=1,nPromedios
        DO  I=1,N
        cEH(I,I)=(0.D0,0.D0)
```

```
          DO  J=I+1,N
          cEH(I,J)=(0.D0,0.D0)
          cEH(J,I)=(0.D0,0.D0)
          enddo
        enddo

C*********************************************************


C*********************************************************
C-defino las autoenergias y los hoping efectivos (prcedimiento de decimation)
c        ________________________________
c        inicializo variables del Hamiltoniano efectivo
c       SR=0, SL=0, Veff=Vx*rUNIT

        do i=1,N
         do j=1,N
          cSR(i,j)=(0.d0,0.d0)
          cSL(i,j)=(0.d0,0.d0)
          cVeff(i,j)=(0.d0,0.d0)
         enddo
          cVeff(i,i)=Vx*cRE
        end do
C       --------------------------------
c       comienza eliminacion de (Lm-1) capas internas del sistema

        if (Lm1.ge.1) then
         do it=1,Lm1

               cHOP=Vy
               cHOPc=dconjg(cHOP)

        do I=1,N
               Iu=I-1
               cEH(I,I)=(Ef-W*(RANDU(SEM1)-.5))*cRE !+(0.0001*cIM)
               If(I.GT.1) then
                      cEH(I,Iu)=-cHOP
                      cEH(Iu,I)=-cHOPC
               end if
        enddo !I

             call cSUM(Gi,1.d0,cEH,-1.d0,cSR,N)
             call cINV(Gi,N)

c            *****SL=SL+Veff(1/(E-Ei-SR))Veff*****
             call cPROD(cSwap,1.d0,cVeff,Gi,N)
             call cTRAN(ctVeff,cVeff,N)
             call cPROD(cSR,1.d0,cSwap,ctVeff,N)
             call cSUM(cSL,1.d0,cSL,1.d0,cSR,N)

c            ******SR=(V(i+1)**2)/(E-Ei-SR)*******
              call cEQUAL(cSR,Vx**2,Gi,N)

c            ******Veff=Veff*1/(E-Ei-SR)*V(i+1)******
             call cEQUAL(cVeff,Vx,cSwap,N)

          end do !it
         end  if !si el cuadro es 2x2 salta estas operaciones.

c Defino las autoenergias de los alambres externos (en la base k)

         do k=1,N
         cSk(k)=cS( Ef-2.D0*Vy*dcos(k*rPI/Np) , Vx )
         end do

c       defino la matriz de las autoenergias de los alambres en la base de sitio
         do I1=1,N
          do I2=1,N
          ctmp=(0.d0,0.d0)
           do k=1,N
         ctmp=ctmp+(2.D0/Np)*dsin(I1*k*rPI/Np)*dsin(I2*k*rPI/Np)*cSk(k)
            end do!k
          cSI(I1,I2)=ctmp
         end do !I2
        end do !I1

C Defino el Hamiltoniano efectivo escrito en la base de la primera y
C ultima capa del sistema teniendo en cuenta los alambres
```

```
c     ___________Comienzo definiendo las contribuciones
C    del sistema y las decimaciones de los alambres
       do i=1,N
           do j=1,N
c            Pone la contribucion de los sitios eliminados
           cEHeff(i,j)=    -cSL(i,j) -cSI(i,j)
           cEHeff(N+i,N+j)=-cSR(i,j) -cSI(i,j)
           cEHeff(i,N+j)=      -cVeff(i,j)
           cEHeff(N+j,i)=  -dconjg(cVeff(i,j)) !correct because eta=0 real
           end do !j

c     Aagrego las energias diagonales no perturbadas de la capa
          cEHeff(i,i)=cEHeff(i,i)+(Ef-W*(RANDU(SEMI)-.5))*cRE
          cEHeff(N+i,N+i)=cEHeff(N+i,N+i)+(Ef-W*(RANDU(SEMI)-.5))*cRE
       enddo !i

       do i=1,N-1
c     Agrego los hopings en la capa
       i2=i+1
                     cEHeff(i,i2)=cEHeff(i,i2)-Vy
                     cEHeff(i2,i)=cEHeff(i2,i)-Vy
                     cEHeff(N+i,N+i2)=cEHeff(N+i,N+i2)-Vy
                     cEHeff(N+i2,N+i)=cEHeff(N+i2,N+i)-Vy
       end do   ! i

       call cEQUAL(cGLR,1.d0,cEHeff,N2)

       call cINV(cGLR,N2)

c ***********************************************************
c Transformo la funcion de Green a la base k
       do k1=1,N
       do k2=1,N
       ct=(0.D0,0.D0)
       c2=(0.D0,0.D0)
         do I1=1,N
         do I2=1,N
       ct=ct+(2.D0/Np)*dsin(I1*k1*rPI/Np)*cGLR(I1,N+I2)
       1                    *dsin(I2*k2*rPI/Np)
       c2=c2+(2.D0/Np)*dsin(I1*k1*rPI/Np)*cGLR(I1,I2)*dsin(I2*k2*rPI/Np)
        end do
        end do
       cG0L(k1,k2)=ct
       cG00(k1,k2)=c2
       end do
       enddo

c***Calculo de la funcion de Green del sistema

       rgt=0.D0
       rr=0.d0
       Do k1=1,N
              cGreen(k1)=cGreen(k1)+(cG0L(k1,k1))/nPromedios
       end do !k1

9999   continue
C     Este es el bucle de los promedios
c      Cuando termino de promediar, grabo para esa energia

       rprom=0
       ncanales=0
       do k1=1,N
              rtemp=cdabs(cGreen(k1))
              if (rtemp.eq.0) then
                     rtemp=-1
              else
C             Calculo del camino libre medio segun definicion de Abrikosov,
C             si es menor a 1 se reemplaza por 1
                     rtemp=-log(rtemp/rmodGo(nEf+1,k1))/Lm1
           endif
              if ((rtemp.le.0).or.(rtemp.gt.1))    then
                     rmodG(k1)=1
                     rtemp=501.d0
              else
                     rtemp=1/rtemp
              endif
              if (rtemp.gt.500) then
                     rmodG(k1)=1
              else
                     rmodG(k1)=rtemp
```

```
               rprom=rprom+rmodG(k1)
                     ncanales=ncanales+1
              endif
       enddo
       if (ncanales.eq.0) then
              rprom=1
              rsuma=1
       else

C      Se corrije el camino libre medio para la cantidad
C      de canales abiertos a esa energia
              rprom=rprom*(ncanales+1)/(2*ncanales**2)
              rsuma=rprom*ncanales
       endif

      open (11,FILE=trim(dir)//trim(filename)//'.dat',STATUS='UNKNOWN'
       1   ,POSITION='APPEND')

c      Salida: Energia de Fermi, l, Ma l
       WRITE(11,'(41G22.15)') (Ef+Vy),rprom ,rsuma

       close(11)
1000   CONTINUE ! va a buscar un nuevo parametro
       END

C-----------------------------------------------------------------
C-----------       FUNCIONES COMPLEMENTARIAS       ---------------
C-----------------------------------------------------------------

       real*8 FUNCTION RANDU(DSEED)
       real*8 DSEED
       real*8 D2P31M, D2P31
       DATA  D2P31M/2147483647.D0/,D2P31/2147483649.D0/
       DSEED=DMOD(16807.D0*DSEED,D2P31M)
       RANDU=DSEED/D2P31
       RETURN
       END

       complex*16 function cS(E,V)
       real*8 E,rdisc,V
       complex*16 cRE, cIM
       cRE=(1.D0,0.D0)
       cIM=(0.D0,1.D0)
       rdisc=(E/2.)**2-V**2
       if(rdisc.gt.0.) then
        cS=( E/2.-(E/dabs(E))*dsqrt(rdisc))*cRE
       else
       cS=E/2.*cRE-cIM*dsqrt(-rdisc)
       endif
       return
       end

      SUBROUTINE INV(A,N)
      PARAMETER (NMAX=50)
      REAL*8 DUM, BIG, PIVINV,A(N,N)
      DIMENSION IPIV(NMAX),INDXR(NMAX),INDXC(NMAX)

      DO 11 J=1,N
        IPIV(J)=0
11    CONTINUE
      DO 22 I=1,N
        BIG=0.
        DO 13 J=1,N
          IF(IPIV(J).NE.1)THEN
            DO 12 K=1,N
              IF (IPIV(K).EQ.0) THEN
                IF (ABS(A(J,K)).GE.BIG)THEN
                  BIG=ABS(A(J,K))
                  IROW=J
                  ICOL=K
                ENDIF
              ELSE IF (IPIV(K).GT.1) THEN
                PAUSE 'Singular matrix'
              ENDIF
12          CONTINUE
          ENDIF
13      CONTINUE
        IPIV(ICOL)=IPIV(ICOL)+1
        IF (IROW.NE.ICOL) THEN
          DO 14 L=1,N
```

```
            DUM=A(IROW,L)
            A(IROW,L)=A(ICOL,L)
            A(ICOL,L)=DUM
14        CONTINUE
        ENDIF
        INDXR(I)=IROW
        INDXC(I)=ICOL
        IF (A(ICOL,ICOL).EQ.0.) PAUSE 'Singular matrix.'
        PIVINV=1./A(ICOL,ICOL)
        A(ICOL,ICOL)=1.
        DO 16 L=1,N
          A(ICOL,L)=A(ICOL,L)*PIVINV
16      CONTINUE
        DO 21 LL=1,N
          IF(LL.NE.ICOL)THEN
            DUM=A(LL,ICOL)
            A(LL,ICOL)=0.
            DO 18 L=1,N
              A(LL,L)=A(LL,L)-A(ICOL,L)*DUM
18          CONTINUE
          ENDIF
21      CONTINUE
22    CONTINUE
      DO 24 L=N,1,-1
        IF(INDXR(L).NE.INDXC(L))THEN
          DO 23 K=1,N
            DUM=A(K,INDXR(L))
            A(K,INDXR(L))=A(K,INDXC(L))
            A(K,INDXC(L))=DUM
23        CONTINUE
        ENDIF
24    CONTINUE

      RETURN
      END

      subroutine cINV(cA,N)
      PARAMETER (NMAX=4000)
      IMPLICIT COMPLEX*16 (C)
      implicit real*8 (a-b,d-h,o-z)
      DIMENSION cA(N,N),IPIV(NMAX),INDXR(NMAX),INDXC(NMAX)
       cRE=(1.d0,0.d0)

      DO 11 J=1,N
        IPIV(J)=0
11    CONTINUE
      DO 22 I=1,N
        BIG=0.d0
        DO 13 J=1,N
          IF(IPIV(J).NE.1)THEN
            DO 12 K=1,N
              IF (IPIV(K).EQ.0) THEN
                IF (cdABS(cA(J,K)).GE.BIG)THEN
                  BIG=cdABS(cA(J,K))
                  IROW=J
                  ICOL=K
                ENDIF
              ELSE IF (IPIV(K).GT.1) THEN
                PAUSE 'Singular matrix'
              ENDIF
12          CONTINUE
          ENDIF
13      CONTINUE
        IPIV(ICOL)=IPIV(ICOL)+1
        IF (IROW.NE.ICOL) THEN
          DO 14 L=1,N
            cDUM=cA(IROW,L)
            cA(IROW,L)=cA(ICOL,L)
            cA(ICOL,L)=cDUM
14        CONTINUE
         ENDIF
        INDXR(I)=IROW
        INDXC(I)=ICOL
        IF (cdabs(cA(ICOL,ICOL)).lt.1.d-107) PAUSE 'Singular matrix.'
        cPIVINV=cRE/cA(ICOL,ICOL)
        cA(ICOL,ICOL)=cRE
        DO 16 L=1,N
          cA(ICOL,L)=cA(ICOL,L)*cPIVINV
16      CONTINUE
        DO 21 LL=1,N
```

```
          IF(LL.NE.ICOL)THEN
            cDUM=cA(LL,ICOL)
            cA(LL,ICOL)=0.*cRE
            DO 18 L=1,N
              cA(LL,L)=cA(LL,L)-cA(ICOL,L)*cDUM
18          CONTINUE

          ENDIF
21      CONTINUE
22    CONTINUE
      DO 24 L=N,1,-1
        IF(INDXR(L).NE.INDXC(L))THEN
          DO 23 K=1,N
            cDUM=cA(K,INDXR(L))
            cA(K,INDXR(L))=cA(K,INDXC(L))
            cA(K,INDXC(L))=cDUM
23        CONTINUE
        ENDIF
24    CONTINUE
      RETURN
      END

c*************************************************************
       subroutine cPROD(cD,r1,cA,cB,n)
C       product of two square matrices
C       D= r1* A* B
       complex*16 cA(n,n),cB(n,n),cD(n,n)
       real*8 r1
       do 3 i=1,n
         do 2 j=1,n
         cD(i,j)=(0.d0,0.d0)
         do 1 k=1,n
  1       cD(i,j)= cD(i,j)+r1*cA(i,k)*cB(k,j)
  2    continue
  3    continue
       return
       end

        subroutine cSUM(D,a1,A,a2,B,n)
C       evaluates the sum of two matrices
C       a1*A + a2* B= D
       complex*16 A(n,n),B(n,n), D(n,n)
       real*8 a1,a2
       do 2 i=1,n
         do 1 j=1,n
  1       D(i,j)=a1*A(i,j)+a2*B(i,j)
  2    continue
       return
       end

       subroutine cTRAN(ctV,cV,N)
c         evaluates the transpose conjugate
        complex*16 ctV(N,N),cV(N,N)
        do i=1,N
        do j=1,N
        ctV(i,j)=dconjg(cV(j,i))
       end do
       enddo
       return
       end

       subroutine cEQUAL(cD, r1,cA,n)
c       evaluates cD= r1* cA
       complex*16 cA(n,n),cD(n,n)
       real*8 r1
       do 2 i=1,n
         do 1 j=1,n
   1    cD(i,j)= r1* cA(i,j)
   2   continue
       return
       end
```

```
c       Apéndice al capitulo 3
~
c       Programa para calcular el camino libre medio de un Hamiltoniano
~
C        en el sistema estrella
~

~
c  ************************
~
        program LMEDIO
~
        use portlib

C-------------------------------------------------------
C       Ef = Energia de Fermi investigada
C       N  = Numero de canales (sitios por capa)
C       Vx = Hoping entre capas
C       Vy = Hoping en la capa
C       cSL , cSR matrices complejas con las renormalizaciones
C              de los alambres
C       E_Hef  = Energia de Fermi menos el Hamiltoniano Efectivo
C       gi             =   Localizador de capa renormalizado
C       cGLR   = Matriz de la Funcion de Green final
C       rmodGo  = Funcion de Green para el sistema ordenado
c*************************************************************
        IMPLICIT REAL*8 (E,F,O-Y)
         IMPLICIT COMPLEX*16 (A-d,G-H)
        IMPLICIT INTEGER (I-N)

        PARAMETER (rPI=3.1415926535897932D0)
        PARAMETER (N=6)
        parameter(N2=N*2)
        PARAMETER (NF=500)
        character(60) filename, dir

         DIMENSION  cSR(N,N), cSL(N,N), Gi(N,N),cSwap(N,N)
     1     ,cVeff(N,N),ctVeff(N,N),rmodG(N)
     2   , cEHeff(N2,N2),cGLR(N2,N2),cSI(N,N),rmodGo(NF,N)
     3   ,cEH(N,N),cG0L(N,N),cG00(N,N),cSk(N),rT(4,4),cGreen(N)

        double precision RANDU
        complex*16 cS

        dir='c:\datos\'

        write(6,*) 'Ingrese nombre del archivo de salida:'
        read (5,*) filename

        WRITE(6,*)'Emin Emax?, DESORDEN?, Largo, Promeds?'
        READ(5,*)Fmin, Fmax, W, Lm1, nPromedios

        L=Lm1+1
        Np=N
        Fc=0

        open (9,FILE=trim(dir)//trim(filename)//'.log',STATUS='UNKNOWN')

        WRITE(9,*) 'Calculo de l medio en una rueda'
        WRITE(9,*) 'Ancho(Canales:M) ',N
        WRITE(9,*) 'Largo de la rueda (r)',Lm1
       WRITE(9,*) 'Flujo', Fc
        WRITE(9,*) 'Desorden ', W
         WRITE(9,*) 'Energia Min', Fmin
        WRITE (9,*) 'Energia Max', Fmax
        WRITE(9,*) 'Numero de energias ',NF
        WRITE(9,*) 'Numero de Promedios ',nPromedios
        WRITE(9,*) 'Salida:'
        WRITE(9,*) '(Efermi+1/M**0.5) <l>medio  <l>suma'

         close(9)

        Vx=1.D0
        Vy=1/N**0.5

         cIM=(0.D0,1.D0)
~
         cRE=(1.D0,0.D0)
~
```

```
~
       Fmin=Fmin-Vy
~
       Fmax=Fmax-Vy
~

~
c-------------------------------------------------------------
~
c  CALCULO DEL MODULO DE G PARA EL SISTEMA ORDENADO
~
C-------------------------------------------------------------
       ntime1=time()
       W2=W
       W=0
      DO 1999 nEf=0,NF-1
       Ef=(nEf*(Fmax-Fmin)/NF)+Fmin

       cGreen=cGreen*0
       print *,( (100*nEf)/NF), ' % ',(time()-ntime1),' s'

      DO  I=1,N
        cEH(I,I)=(0.D0,0.D0)

         DO  J=I+1,N
        cEH(I,J)=(0.D0,0.D0)
        cEH(J,I)=(0.D0,0.D0)
        enddo
       enddo
C*********************************************************
C-defino las autoenergias y los hoping efectivos (prcedimiento de decimation)
c       ______________________________
c       inicializo variables del Hamiltoniano efectivo
c       SR=0, SL=0, Veff=Vx*rUNIT

       do i=1,N
        do j=1,N
         cSR(i,j)=(0.d0,0.d0)
         cSL(i,j)=(0.d0,0.d0)
         cVeff(i,j)=(0.d0,0.d0)
        enddo
         cVeff(i,i)=Vx*cRE
       end do
C       -------------------------------
c      comienza eliminacion de (Lm-1) capas internas del sistema

       if (Lm1.ge.1) then
        do it=1,Lm1 !toma cada capa

              cHOP=Vy
              cHOPc=dconjg(cHOP)

              do I=1,N
                     cEH(I,I)=(Ef)*cRE
               If(I.LT.N) then
                     do Iu=I+1,N
                            cEH(I,Iu)=-cHOP
                            cEH(Iu,I)=-cHOPC
                     enddo !Iu
                     endif
              enddo !I

            call cSUM(Gi,1.d0,cEH,-1.d0,cSR,N)
            call cINV(Gi,N)

c           *****SL=SL+Veff(1/(E-Ei-SR))Veff*****
            call cPROD(cSwap,1.d0,cVeff,Gi,N)
            call cTRAN(ctVeff,cVeff,N)
            call cPROD(cSR,1.d0,cSwap,ctVeff,N)
            call cSUM(cSL,1.d0,cSL,1.d0,cSR,N)

c           ******SR=(V(i+1)**2)/(E-Ei-SR)*******
             call cEQUAL(cSR,Vx**2,Gi,N)

c           ******Veff=Veff*1/(E-Ei-SR)*V(i+1)******
            call cEQUAL(cVeff,Vx,cSwap,N)
         end do
        end  if !si el cuadro es 2x2 salta estas operaciones.
```

```
c Defino las autoenergias de los alambres externos (en la base k)

       do k=1,N
              rEne=0
              do i1=1,N-1
                     rEne=rEne+Vy*cdexp(cIM*2*i1*(k-1)*rPI/Np)
              enddo
              cSk(k)=cS( Ef-rEne , Vx )
       enddo
c      defino la matriz de las autoenergias de los alambres en la base de sitio
        do I1=1,N
         do I2=1,N
         ctmp=(0.d0,0.d0)
          do k=1,N
        ctmp=ctmp+(1.D0/Np)*cdexp(cIM*2*(I1-1)*(k-1)*rPI/Np)*cSk(k)
       1                    *cdexp(-cIM*2*(I2-1)*(k-1)*rPI/Np)
          end do!k
         cSI(I1,I2)=ctmp
        end do !I2
       end do !I1

C Defino el Hamiltoniano efectivo escrito en la base de la primera y
C ultima capa del sistema teniendo en cuenta los alambres

c     ___________Comienzo definiendo las contribuciones
C    del sistema y las decimaciones de los alambres
       do i=1,N
           do j=1,N
c            Pone la contribucion de los sitios eliminados
           cEHeff(i,j)=    -cSL(i,j) -cSI(i,j)
           cEHeff(N+i,N+j)=-cSR(i,j) -cSI(i,j)
           cEHeff(i,N+j)=      -cVeff(i,j)
           cEHeff(N+j,i)=  -dconjg(cVeff(i,j))
           end do !j

c     Aagrego las energias diagonales no perturbadas de la capa
          cEHeff(i,i)=cEHeff(i,i)+(Ef)*cRE
          cEHeff(N+i,N+i)=cEHeff(N+i,N+i)+(Ef)*cRE
       enddo !i
       do i=1,N-1 !
c     Agrego los hopings en la capa
              do i2=i+1,N
                     cEHeff(i,i2)=cEHeff(i,i2)-Vy
                     cEHeff(i2,i)=cEHeff(i2,i)-Vy
                     cEHeff(N+i,N+i2)=cEHeff(N+i,N+i2)-Vy
                     cEHeff(N+i2,N+i)=cEHeff(N+i2,N+i)-Vy
              enddo
       end do   ! i

       call cEQUAL(cGLR,1.d0,cEHeff,N2)
       call cINV(cGLR,N2)
c Transformo la funcion de Green a la base k
       do k1=1,N
       do k2=1,N
       ct=(0.D0,0.D0)
       c2=(0.D0,0.D0)
         do I1=1,N
         do I2=1,N
       ct=ct+(1.D0/Np)*cdexp(-cIM*2*(I1-1)*(k1-1)*rPI/Np)*cGLR(I1,N+I2)
       1                *cdexp(cIM*2*(I2-1)*(k2-1)*rPI/Np)
       c2=c2+(1.D0/Np)*cdexp(-cIM*2*(I1-1)*(k1-1)*rPI/Np)*cGLR(I1,I2)
       1                *cdexp(cIM*2*(I2-1)*(k2-1)*rPI/Np)
        end do
        end do
       cG0L(k1,k2)=ct
       cG00(k1,k2)=c2
       end do
       enddo

c***Calculo del modulo de la funcion de Green para
c el sistema ordenado, Go

       Do k1=1,N
      rmodGo(nEf+1,k1)=cdabs(cG0L(k1,k1))
       if (rmodGo(nEf+1,k1).eq.0) rmodGo(nEf+1,k1)=10**(-15)
       end do !k1



1999   Continue
```

```
c---------------------------------------------------------------
c  --------------Comienza el calculo del camino libre medio-------


        W=W2
        ntime1=time()

       DO 1000 nEf=0,NF-1
        Ef=(nEf*(Fmax-Fmin)/NF)+Fmin
        cGreen=cGreen*0
        print *,( (100*(Ef-Fmin))/(Fmax-Fmin)), ' % ',(time()-ntime1),' s'
        SEM1=.23821

        do 9999 nprom=1,nPromedios
         DO  I=1,N
         cEH(I,I)=(0.D0,0.D0)

           DO  J=I+1,N
          cEH(I,J)=(0.D0,0.D0)
          cEH(J,I)=(0.D0,0.D0)
          enddo
        enddo



C*********************************************************


C*********************************************************
C-defino las autoenergias y los hoping efectivos (prcedimiento de decimation)
c        ______________________________
c        inicializo variables del Hamiltoniano efectivo
c        SR=0, SL=0, Veff=Vx*rUNIT

        do i=1,N
         do j=1,N
          cSR(i,j)=(0.d0,0.d0)
          cSL(i,j)=(0.d0,0.d0)
          cVeff(i,j)=(0.d0,0.d0)
         enddo
          cVeff(i,i)=Vx*cRE
        end do
C       --------------------------------


C       --------------------------------
c       comienza eliminacion de (Lm-1) capas internas del sistema

        if (Lm1.ge.1) then
         do it=1,Lm1

                cHOP=Vy
                cHOPc=dconjg(cHOP)

        do I=1,N
         cEH(I,I)=(Ef-W*(RANDU(SEM1)-.5))*cRE
         If(I.LT.N) then
                do Iu=I+1,N
                        cEH(I,Iu)=-cHOP
                        cEH(Iu,I)=-cHOPC
                enddo
        end if
         enddo !I

             call cSUM(Gi,1.d0,cEH,-1.d0,cSR,N)
             call cINV(Gi,N)

c            *****SL=SL+Veff(1/(E-Ei-SR))Veff*****
             call cPROD(cSwap,1.d0,cVeff,Gi,N)
             call cTRAN(ctVeff,cVeff,N)
             call cPROD(cSR,1.d0,cSwap,ctVeff,N)
             call cSUM(cSL,1.d0,cSL,1.d0,cSR,N)

c            ******SR=(V(i+1)**2)/(E-Ei-SR)*******
              call cEQUAL(cSR,Vx**2,Gi,N)

c            ******Veff=Veff*1/(E-Ei-SR)*V(i+1)******
             call cEQUAL(cVeff,Vx,cSwap,N)
```

```
         end do !it
        end  if !si el cuadro es 2x2 salta estas operaciones.

c Defino las autoenergias de los alambres externos (en la base k)

       do k=1,N
              rEne=0
              do i1=1,N-1
                     rEne=rEne+Vy*cdexp(cIM*2*i1*(k-1)*rPI/Np)
              enddo
              cSk(k)=cS( Ef-rEne , Vx )
       enddo


c      defino la matriz de las autoenergias de los alambres en la base de sitio
        do I1=1,N
         do I2=1,N
         ctmp=(0.d0,0.d0)
          do k=1,N
        ctmp=ctmp+(1.D0/Np)*cdexp(cIM*2*(I1-1)*(k-1)*rPI/Np)*cSk(k)
       1                     *cdexp(-cIM*2*(I2-1)*(k-1)*rPI/Np)
          end do!k
         cSI(I1,I2)=ctmp
        end do !I2
       end do !I1


C Defino el Hamiltoniano efectivo escrito en la base de la primera y
C ultima capa del sistema teniendo en cuenta los alambres

c     ___________Comienzo definiendo las contribuciones
C    del sistema y las decimaciones de los alambres
       do i=1,N
           do j=1,N
c            Pone la contribucion de los sitios eliminados
           cEHeff(i,j)=    -cSL(i,j) -cSI(i,j)
           cEHeff(N+i,N+j)=-cSR(i,j) -cSI(i,j)
           cEHeff(i,N+j)=      -cVeff(i,j)
           cEHeff(N+j,i)=  -dconjg(cVeff(i,j)) !correct because eta=0 real
           end do !j

c     Aagrego las energias diagonales no perturbadas de la capa
          cEHeff(i,i)=cEHeff(i,i)+(Ef-W*(RANDU(SEMI)-.5))*cRE
          cEHeff(N+i,N+i)=cEHeff(N+i,N+i)+(Ef-W*(RANDU(SEMI)-.5))*cRE
       enddo !i

       do i=1,N-1
c     Agrego los hopings en la capa
              do i2=i+1,N
                     cEHeff(i,i2)=cEHeff(i,i2)-Vy
                     cEHeff(i2,i)=cEHeff(i2,i)-Vy
                     cEHeff(N+i,N+i2)=cEHeff(N+i,N+i2)-Vy
                     cEHeff(N+i2,N+i)=cEHeff(N+i2,N+i)-Vy
              enddo
       end do   ! i


       call cEQUAL(cGLR,1.d0,cEHeff,N2)


       call cINV(cGLR,N2)



c ********************************************************



c Transformo la funcion de Green a la base k
       do k1=1,N
       do k2=1,N
       ct=(0.D0,0.D0)
       c2=(0.D0,0.D0)
         do I1=1,N
         do I2=1,N
       ct=ct+(1.D0/Np)*cdexp(-cIM*2*(I1-1)*(k1-1)*rPI/Np)*cGLR(I1,N+I2)
       1               *cdexp(cIM*2*(I2-1)*(k2-1)*rPI/Np)
       c2=c2+(1.D0/Np)*cdexp(-cIM*2*(I1-1)*(k1-1)*rPI/Np)*cGLR(I1,I2)
```

```
       1               *cdexp(cIM*2*(I2-1)*(k2-1)*rPI/Np)
        end do
        end do
       cG0L(k1,k2)=ct
       cG00(k1,k2)=c2
       end do
       enddo

c***Calculo de la funcion de Green del sistema

       rgt=0.D0
       rr=0.d0
       Do k1=1,N
              cGreen(k1)=cGreen(k1)+(cG0L(k1,k1))/nPromedios
       end do !k1

9999   continue
C     Este es el bucle de los promedios
c      Cuando termino de promediar, grabo para esa energia

       rprom=0
       ncanales=0
       do k1=1,N
              rtemp=cdabs(cGreen(k1))
              if (rtemp.eq.0) then
                     rtemp=-1
              else
C             Calculo del camino libre medio segun definicion de Abrikosov,
C             si es menor a 1 se reemplaza por 1
                     rtemp=-log(rtemp/rmodGo(nEf+1,k1))/Lm1
           endif
              if ((rtemp.le.0).or.(rtemp.gt.1))    then
                     rmodG(k1)=1
                     rtemp=501.d0
              else
                     rtemp=1/rtemp
              endif
              if (rtemp.gt.500) then
                     rmodG(k1)=1
              else
                     rmodG(k1)=rtemp
               rprom=rprom+rmodG(k1)
                     ncanales=ncanales+1
              endif
       enddo
       if (ncanales.eq.0) then
              rprom=1
              rsuma=1
       else

C      Se corrije el camino libre medio para la cantidad
C      de canales abiertos a esa energia
              rprom=rprom*(ncanales+1)/(2*ncanales**2)
              rsuma=rprom*ncanales
       endif

      open (11,FILE=trim(dir)//trim(filename)//'.dat',STATUS='UNKNOWN'
       1   ,POSITION='APPEND')

c      Salida: Energia de Fermi, l, Ma l
       WRITE(11,'(41G22.15)') (Ef+Vy),rprom ,rsuma

       close(11)
1000   CONTINUE ! va a buscar un nuevo parametro
        END

C---------------------------------------------------------------------
C-----------       FUNCIONES COMPLEMENTARIAS       ---------------
C---------------------------------------------------------------------

       real*8 FUNCTION RANDU(DSEED)
       real*8 DSEED
       real*8 D2P31M, D2P31
       DATA  D2P31M/2147483647.D0/,D2P31/2147483649.D0/
       DSEED=DMOD(16807.D0*DSEED,D2P31M)
       RANDU=DSEED/D2P31
       RETURN
       END

       complex*16 function cS(E,V)
```

```
       real*8 E,rdisc,V
       complex*16 cRE, cIM
       cRE=(1.D0,0.D0)
       cIM=(0.D0,1.D0)
       rdisc=(E/2.)**2-V**2
       if(rdisc.gt.0.) then
        cS=( E/2.-(E/dabs(E))*dsqrt(rdisc))*cRE
       else
       cS=E/2.*cRE-cIM*dsqrt(-rdisc)
       endif
       return
       end


      SUBROUTINE INV(A,N)
      PARAMETER (NMAX=50)
      REAL*8 DUM, BIG, PIVINV,A(N,N)
      DIMENSION IPIV(NMAX),INDXR(NMAX),INDXC(NMAX)


      DO 11 J=1,N
        IPIV(J)=0
11    CONTINUE
      DO 22 I=1,N
        BIG=0.
        DO 13 J=1,N
          IF(IPIV(J).NE.1)THEN
            DO 12 K=1,N
              IF (IPIV(K).EQ.0) THEN
                IF (ABS(A(J,K)).GE.BIG)THEN
                  BIG=ABS(A(J,K))
                  IROW=J
                  ICOL=K
                ENDIF
              ELSE IF (IPIV(K).GT.1) THEN
                PAUSE 'Singular matrix'
              ENDIF
12          CONTINUE
          ENDIF
13      CONTINUE
        IPIV(ICOL)=IPIV(ICOL)+1
        IF (IROW.NE.ICOL) THEN
          DO 14 L=1,N
            DUM=A(IROW,L)
            A(IROW,L)=A(ICOL,L)
            A(ICOL,L)=DUM
14        CONTINUE
        ENDIF
        INDXR(I)=IROW
        INDXC(I)=ICOL
        IF (A(ICOL,ICOL).EQ.0.) PAUSE 'Singular matrix.'
        PIVINV=1./A(ICOL,ICOL)
        A(ICOL,ICOL)=1.
        DO 16 L=1,N
          A(ICOL,L)=A(ICOL,L)*PIVINV
16      CONTINUE
        DO 21 LL=1,N
          IF(LL.NE.ICOL)THEN
            DUM=A(LL,ICOL)
            A(LL,ICOL)=0.
            DO 18 L=1,N
              A(LL,L)=A(LL,L)-A(ICOL,L)*DUM
18          CONTINUE
          ENDIF
21      CONTINUE
22    CONTINUE
      DO 24 L=N,1,-1
        IF(INDXR(L).NE.INDXC(L))THEN
          DO 23 K=1,N
            DUM=A(K,INDXR(L))
            A(K,INDXR(L))=A(K,INDXC(L))
            A(K,INDXC(L))=DUM
23        CONTINUE
        ENDIF
24    CONTINUE


      RETURN
      END
```

```
      subroutine cINV(cA,N)
      PARAMETER (NMAX=4000)
      IMPLICIT COMPLEX*16 (C)
      implicit real*8 (a-b,d-h,o-z)
      DIMENSION cA(N,N),IPIV(NMAX),INDXR(NMAX),INDXC(NMAX)
       cRE=(1.d0,0.d0)

      DO 11 J=1,N
        IPIV(J)=0
11    CONTINUE
      DO 22 I=1,N
        BIG=0.d0
        DO 13 J=1,N
          IF(IPIV(J).NE.1)THEN
            DO 12 K=1,N
              IF (IPIV(K).EQ.0) THEN
                IF (cdABS(cA(J,K)).GE.BIG)THEN
                  BIG=cdABS(cA(J,K))
                  IROW=J
                  ICOL=K
                ENDIF
              ELSE IF (IPIV(K).GT.1) THEN
                PAUSE 'Singular matrix'
              ENDIF
12          CONTINUE
          ENDIF
13      CONTINUE
        IPIV(ICOL)=IPIV(ICOL)+1
        IF (IROW.NE.ICOL) THEN
          DO 14 L=1,N
            cDUM=cA(IROW,L)
            cA(IROW,L)=cA(ICOL,L)
            cA(ICOL,L)=cDUM
14        CONTINUE
         ENDIF
        INDXR(I)=IROW
        INDXC(I)=ICOL
        IF (cdabs(cA(ICOL,ICOL)).lt.1.d-107) PAUSE 'Singular matrix.'
        cPIVINV=cRE/cA(ICOL,ICOL)
        cA(ICOL,ICOL)=cRE
        DO 16 L=1,N
          cA(ICOL,L)=cA(ICOL,L)*cPIVINV
16      CONTINUE
        DO 21 LL=1,N
          IF(LL.NE.ICOL)THEN
            cDUM=cA(LL,ICOL)
            cA(LL,ICOL)=0.*cRE
            DO 18 L=1,N
              cA(LL,L)=cA(LL,L)-cA(ICOL,L)*cDUM
18          CONTINUE

          ENDIF
21      CONTINUE
22    CONTINUE
      DO 24 L=N,1,-1
        IF(INDXR(L).NE.INDXC(L))THEN
          DO 23 K=1,N
            cDUM=cA(K,INDXR(L))
            cA(K,INDXR(L))=cA(K,INDXC(L))
            cA(K,INDXC(L))=cDUM
23        CONTINUE
        ENDIF
24    CONTINUE
      RETURN
      END

c*************************************************************

       subroutine cPROD(cD,r1,cA,cB,n)
C       product of two square matrices
C       D= r1* A* B
       complex*16 cA(n,n),cB(n,n),cD(n,n)
       real*8 r1
       do 3 i=1,n
         do 2 j=1,n
         cD(i,j)=(0.d0,0.d0)
         do 1 k=1,n
  1       cD(i,j)= cD(i,j)+r1*cA(i,k)*cB(k,j)
```

```
  2     continue
  3     continue
        return
        end


        subroutine cSUM(D,a1,A,a2,B,n)
C       evaluates the sum of two matrices
C       a1*A + a2* B= D
        complex*16 A(n,n),B(n,n), D(n,n)
        real*8 a1,a2
        do 2 i=1,n
          do 1 j=1,n
  1       D(i,j)=a1*A(i,j)+a2*B(i,j)
  2     continue
        return
        end


        subroutine cTRAN(ctV,cV,N)
c         evaluates the transpose conjugate
        complex*16 ctV(N,N),cV(N,N)
        do i=1,N
        do j=1,N
        ctV(i,j)=dconjg(cV(j,i))
        end do
        enddo
        return
        end


        subroutine cEQUAL(cD, r1,cA,n)
c       evaluates cD= r1* cA
        complex*16 cA(n,n),cD(n,n)
        real*8 r1
        do 2 i=1,n
          do 1 j=1,n
   1    cD(i,j)= r1* cA(i,j)
   2    continue
        return
        end
```

```
! Apéndice al capitulo 4
~
! Programa para calcular la evolucion sobre un sistema de dos sitios con cada uno de ellos
~
! acoplado a una cadena, tomando como estado inicial una superposicion de estados
!
!        Vp
!      0---0
! Vs|   |Vs
!      0   0
!  V|  |V
!      0   0
!  V|  |V
!


program decoherencia

use msflib
use msimsl
use portlib

implicit none

!variables para bucles y cuentas intermedias
logical salir
!pongo las variables internas
integer n,n2, time1, time2,p ,k
integer, parameter :: steps=2000
!logical checked
real(8) t, Pi,rtemp
real (8),allocatable :: P11(:),rtiempo(:),Poo(:)
complex(8) comp
character(60) mensaje,ctemp
real(8) , parameter :: hbarra=6.59
!hbarra tiene E-16
complex(8), parameter :: I = (0,1) !unidad imaginaria
character(60) archivodatos ,dir
!Numero total de sitios, Sitios en cada cadena
integer Ntotal, Nmitad
! Energia de cada sitio, Hoping superficial, Hoping en las cadenas
! Hoping entre los primeros sitios
real(8) Eo, Vs,Vc,V
real (8) tiempo, tiempoini
real(4) azar

data Ntotal /200/ , Nmitad /500/
data Eo /0.0/, Vc /0.5/, Vs /0.5/ , V /1/
data tiempo /200/, tiempoini /0.0/
data dir /'c:\datos\'/

!Estos son para H
complex(8),allocatable :: autovals(:)
real(8),allocatable :: H(:,:)
complex(8),allocatable :: ket(:),vector(:),bra(:)
complex(8),allocatable :: invbase(:,:)

salir = .true.
dir = adjustl(dir)
Pi=4*atan(1.0)

principalout: do while (salir)
       write(6,*) 'Ingrese nombre del archivo de salida:'
       read (5,*) archivodatos
       if (archivodatos.eq.'0') then !me fijo si N=0, si es asi pongo salir a .f. y hago un cycle
              salir = .false.
              cycle
       endif
       write(6,*) 'Ingrese largo de la cadena'
       read (5,*) Nmitad
       Ntotal=Nmitad*2

       write(6,*) 'Ingrese tiempo inicial'
       read (5,*) tiempoini
       write(6,*) 'Ingrese tiempo final'
       read (5,*) tiempo
       write(6,*) 'Ingrese hoping Vc'
       read (5,*) Vc
       write(6,*) 'Ingrese hoping superficial Vs'
       read (5,*) Vs
```

```
        write(6,*) 'Ingrese energia de las cadenas'
        read (5,*) Eo


        write(6,*) 'Calculando...'

        !Estos son para H
        allocate (autovals(Ntotal))
        allocate (H(Ntotal,Ntotal))
        allocate (ket(Ntotal),vector(Ntotal))
        allocate (invbase(Ntotal,Ntotal))
        allocate (rtiempo(steps+1),P11(steps+1),Poo(steps+1))
        allocate (bra(Ntotal))
!       call calculos(.true.)
print *, 'Memoria asignada, diagonalizando'

do n=0,steps
rtiempo(n+1) = ((n*(tiempo-tiempoini))/steps + tiempoini)
enddo

time1 = time()

! Inicializo todas las matrices y vectores a cero
H = H*0
P11= P11*0

! lleno el Hamiltoniano de valores
do n=1,Nmitad-1
        H(n+1+Nmitad,n+Nmitad)=Vc
        H(n+Nmitad,n+1+Nmitad)=Vc
        H(n+1,n)=Vc
        H(n,n+1)=Vc
enddo
H(1,2)=Vs
H(2,1)=Vs
H(1+Nmitad,2+Nmitad)=Vs
H(2+Nmitad,1+Nmitad)=Vs

! calculo autovectoresy autovalores
call devcrg (Ntotal, H, Ntotal, autovals, invbase,Ntotal)

print *, 'Diagonalizacion terminada'
open (4, FILE = trim(dir)//trim(archivodatos)//'.avals',ACTION='WRITE', STATUS = 'UNKNOWN')
do n=1,Ntotal
write (4,*) n,' ',dreal(autovals(n)),dreal(invbase(1,n))**2
enddo
close(4)

! ____________________________________________________________
! COMIENZA EL CALCULO DE LA EVOLUCION
!PARA EL ESTADO INICIAL
ket=ket*0
do n = 1,Ntotal
                ket(n) = invbase(1,n) + invbase(1+Nmitad,n)
        enddo
ket=ket/2**.5

print *, 'Multiplicacion terminada - Comienza evolucion'

principal: do n = 0,steps

        !Pongo el tiempo en dimensiones de h/V
        t = rtiempo(n+1)

        !Hago la evolucion en |v>
        interno: do n2 = 1, Ntotal
                vector(n2) = ket(n2) * cdexp(-I* t * autovals(n2))
        enddo interno
        !multiplico sum(i) <bra(i)|vector>
        comp = dot_product (ket, vector)
        Poo(n+1) = comp * dconjg (comp)

        do n2 = 1,Nmitad !Indice de sitio, solo sumo sobre la cadena izq.
                do k=1, Ntotal
                        bra(k)=invbase(n2,k)
                enddo
                comp = dot_product (bra, vector)
                P11(n+1) = comp * dconjg (comp)+ P11(n+1)
        enddo
```

```
end do  principal

! TERMINA LA EVOLUCION
!_____________________________________________________

open (4, FILE = trim(dir)//trim(archivodatos)//'.dat',ACTION='WRITE', STATUS = 'UNKNOWN')
do n=0,steps
! salida: Tiempo, Autocorrelacion al estado inicial, Decoherencia
       write (4,*) rtiempo(n+1),Poo(n+1),P11(n+1)
enddo
close (4)

       !Estos son para H
       deallocate (autovals)
       deallocate (H)
       deallocate (ket,vector)
       deallocate (invbase)
       deallocate (rtiempo,P11,Poo)
       deallocate (bra)


open (4, FILE = trim(dir)//trim(archivodatos)//'.log',ACTION='WRITE', STATUS = 'UNKNOWN')
       write (4,*) 'Dos sitios con cada uno acoplado a cadena, estado inicial como suma de los dos
puntas de cadena'
       write (4,*) 'Formato'
       write (4,*) 't        Poo(t) P11(t)  '
       write (4,*) 'Largo de la cadena     : ', Nmitad
       write (4,*) 'Hoping Vc              : ', Vc
       write (4,*) 'Hoping V entre sitio   : ', V
       write (4,*) 'Hoping Vs entre sitio  : ', Vs
       write (4,*) 'Energia Eo             : ', Eo
       write (4,*) 'Tiempo inicial         : ', tiempoini
       write (4,*) 'Tiempo final           : ', tiempo
       write (4,*) 'Pasos                  : ', steps
       write (4,*) '    Vp  '
       write (4,*) '  0---0   '
       write (4,*) ' Vs|   |Vs       '
       write (4,*) '  0   0    '
       write (4,*) '  V|     |V      '
       write (4,*) '  0   0    '
       write (4,*) '  V|     |V      '

       call time(mensaje)
       call date(ctemp)
       write (4,*) 'Fecha y Hora           : ', trim(ctemp),' ',trim(mensaje)
close (4)


print *, 'Evolucion terminada'

time2 = time()

time1 = time2 - time1

print *, 'Tiempo total [s]' ,time1


enddo  principalout

end program cadenas
```

```
! Apéndice al capitulo 4

! Programa para calcular la evolucion sobre un sistema de dos sitios con uno de ellos
! acoplado a una cadena.
!
!        Vp
!      0---0
!      |Vs
!      0---0
!      |V
!      0---0
!      |V
!


program decoherencia

use msflib
use msimsl
use portlib

implicit none

!variables para bucles y cuentas intermedias
logical salir
!pongo las variables internas
integer n,n2, time1, time2,p ,k
integer, parameter :: steps=2000
!logical checked
real(8) t, Pi,rtemp
real (8),allocatable :: P11(:),rtiempo(:),Poo(:)
complex(8) comp
character(60) mensaje,ctemp
real(8) , parameter :: hbarra=6.59
!hbarra tiene E-16
complex(8), parameter :: I = (0,1) !unidad imaginaria
character(60) archivodatos ,dir
!Numero total de sitios, Sitios en cada cadena
integer Ntotal, Nmitad
! Energia de cada sitio, Hoping superficial, Hoping en las cadenas
! Hoping entre los primeros sitios
real(8) Eo, Vs,Vc,V
real (8) tiempo, tiempoini
real(4) azar

data Ntotal /200/ , Nmitad /500/
data Eo /0.0/, Vc /0.5/, Vs /0.5/ , V /1/
data tiempo /200/, tiempoini /0.0/
data dir /'c:\datos\'/

!Estos son para H
complex(8),allocatable :: autovals(:)
real(8),allocatable :: H(:,:)
complex(8),allocatable :: ket(:),vector(:),bra(:)
complex(8),allocatable :: invbase(:,:)

salir = .true.
dir = adjustl(dir)
Pi=4*atan(1.0)

principalout: do while (salir)
       write(6,*) 'Ingrese nombre del archivo de salida:'
       read (5,*) archivodatos
       if (archivodatos.eq.'0') then !me fijo si N=0, si es asi pongo salir a .f. y hago un cycle
              salir = .false.
              cycle
       endif
       write(6,*) 'Ingrese largo de la cadena'
       read (5,*) Nmitad
       Ntotal=Nmitad*2

       write(6,*) 'Ingrese tiempo inicial'
       read (5,*) tiempoini
       write(6,*) 'Ingrese tiempo final'
       read (5,*) tiempo
       write(6,*) 'Ingrese hoping Vp'
       read (5,*) Vp
       write(6,*) 'Ingrese hoping superficial Vs'
       read (5,*) Vs
       write(6,*) 'Ingrese energia de las cadenas'
```

```
        read (5,*) Eo


        write(6,*) 'Calculando...'

        !Estos son para H
        allocate (autovals(Ntotal))
        allocate (H(Ntotal,Ntotal))
        allocate (ket(Ntotal),vector(Ntotal))
        allocate (invbase(Ntotal,Ntotal))
        allocate (rtiempo(steps+1),P11(steps+1),Poo(steps+1))
        allocate (bra(Ntotal))
!       call calculos(.true.)
print *, 'Memoria asignada, diagonalizando'

do n=0,steps
rtiempo(n+1) = ((n*(tiempo-tiempoini))/steps + tiempoini)
enddo

time1 = time()

! Inicializo todas las matrices y vectores a cero
H = H*0
P11= P11*0

! lleno el Hamiltoniano de valores
do n=1,Nmitad
        H(n,n+Nmitad)=V
        H(n+Nmitad,n)=V
        H(n+1,n)=Vc
        H(n,n+1)=Vc
enddo
H(Nmitad+1,Nmitad)=0
H(Nmitad,Nmitad+1)=0
H(1,2)=Vs
H(2,1)=Vs

! calculo autovectoresy autovalores
call devcrg (Ntotal, H, Ntotal, autovals, invbase,Ntotal)

print *, 'Diagonalizacion terminada'
open (4, FILE = trim(dir)//trim(archivodatos)//'.avals',ACTION='WRITE', STATUS = 'UNKNOWN')
do n=1,Ntotal
write (4,*) n,' ',dreal(autovals(n)),dreal(invbase(1,n))**2
enddo
close(4)

! ____________________________________________________________
! COMIENZA EL CALCULO DE LA EVOLUCION
!PARA EL ESTADO INICIAL
ket=ket*0
do n = 1,Ntotal
                ket(n) = invbase(1,n)
        enddo

print *, 'Multiplicacion terminada - Comienza evolucion'

principal: do n = 0,steps

        !Pongo el tiempo en dimensiones de h/V
        t = rtiempo(n+1)

        !Hago la evolucion en |v>
        interno: do n2 = 1, Ntotal
                vector(n2) = ket(n2) * cdexp(-I* t * autovals(n2))
        enddo interno
        !multiplico sum(i) <bra(i)|vector>
        comp = dot_product (ket, vector)
        Poo(n+1) = comp * dconjg (comp)

        do n2 = 1,Nmitad !Indice de sitio, solo sumo sobre la cadena izq.
                do k=1, Ntotal
                        bra(k)=invbase(n2,k)
                enddo
                comp = dot_product (bra, vector)
                P11(n+1) = comp * dconjg (comp)+ P11(n+1)
        enddo

end do  principal
```

```
! TERMINA LA EVOLUCION
!_____________________________________________________________

open (4, FILE = trim(dir)//trim(archivodatos)//'.dat',ACTION='WRITE', STATUS = 'UNKNOWN')
do n=0,steps
! salida: Tiempo, Autocorrelacion al estado inicial, Decoherencia
       write (4,*) rtiempo(n+1),Poo(n+1),P11(n+1)
enddo
close (4)

       !Estos son para H
       deallocate (autovals)
       deallocate (H)
       deallocate (ket,vector)
       deallocate (invbase)
       deallocate (rtiempo,P11,Poo)
       deallocate (bra)


open (4, FILE = trim(dir)//trim(archivodatos)//'.log',ACTION='WRITE', STATUS = 'UNKNOWN')
       write (4,*) 'Dos sitios con cada uno acoplado a cadena, traza sobre cadena con estado inicial
complejo'
       write (4,*) 'Formato'
       write (4,*) 't        Poo(t) P11(t)  '
       write (4,*) 'Largo de la cadena     : ', Nmitad
       write (4,*) 'Hoping Vc              : ', Vc
       write (4,*) 'Hoping V entre sitio   : ', V
       write (4,*) 'Hoping Vs entre sitio  : ', Vs
       write (4,*) 'Energia Eo             : ', Eo
       write (4,*) 'Tiempo inicial         : ', tiempoini
       write (4,*) 'Tiempo final           : ', tiempo
       write (4,*) 'Pasos                  : ', steps
       call time(mensaje)
       call date(ctemp)
       write (4,*) 'Fecha y Hora           : ', trim(ctemp),' ',trim(mensaje)
close (4)


print *, 'Evolucion terminada'

time2 = time()

time1 = time2 - time1

print *, 'Tiempo total [s]' ,time1


enddo  principalout

end program cadenas
```

```
C       Apéndice al capitulo 4
~
~
C       Calculo de la transmision efectiva de una muestra mediante el
~
C       calculo de los potenciales quimicos de los alambres inelasticos
~
~
        IMPLICIT COMPLEX*16 (C)
        IMPLICIT REAL*8 (A-B,D-H,O-Z)

        DIMENSION CG(402,402),CSR(402),CSL(402),TT(402,402),EI(402)

        OPEN(10,NAME='TEFANT.DAT',TYPE='NEW')

        WRITE (6,*) 'ENERGIA DE FERMI EFI INICIAL'
        READ (5,*) EFI
        WRITE (6,*) 'ENERGIA DE FERMI EFF FINAL'
        READ (5,*) EFF
        WRITE (6,*) 'ENERGIA DE FERMI  PASO EFP'
        READ (5,*) EFP
        WRITE (6,*) 'PARAMETRO DE DESORDEN W'
        READ (5,*) W
        WRITE (6,*) 'NUMERO DE SITIOS DESORDENADOS <=400'
        READ (5,*) M
        WRITE (6,*) 'PARAMETRO INELASTICO VINE'
        READ (5,*) VINE


        CR=(1.D+00,0.D+00)
        CI=(0.D+00,1.D+00)
        CO=(0.D+00,0.D+00)
        V=-1.D+00
        CSI=-CI*VINE
        RVI=2.*VINE

        DO 1000 EF=EFI,EFF,EFP


        RVK=DABS(2.*DSIN(DACOS(EF/2.)))
        CS=EF/2.*CR-CI*DSQRT(1-(EF/2.)**2)


        IS1=32486
        IS2=75939



        EI(1)=0.D+00
        EI(M+2)=0.D+00
        DO I=2,M+1
        EI(I)=(RAN(IS1,IS2)-0.5)*W
        END DO


        CSR(M+2)=CS
        CSR(M+1)=CS
        CSL(1)=CS
        CSL(2)=CS

        DO I=M,1,-1
        CSR(I)=1./((EF-EI(I+1))*CR-CSR(I+1)-CSI)
        END DO

        DO I=3,M+2
        CSL(I)=1./((EF-EI(I-1))*CR-CSL(I-1)-CSI)
        END DO


        CG(1,1)=1./(EF*CR-CSR(1)-CS)
        CG(M+2,M+2)=1./(EF*CR-CSL(M+2)-CS)

        DO I=2,M+1
        CG(I,I)=1./((EF-EI(I))*CR-CSR(I)-CSL(I)-CSI)
        END DO

        DO I=2,M+2
        DO J=1,I-1
```

```
        CG(J,I)=CG(J,I-1)*CSR(I-1)
        END DO
        END DO

        TT(1,1)=(CDABS(CI*RVK*CG(1,1)-1.))**2
        TT(1,M+2)=(CDABS(CG(1,M+2)))**2*RVK**2
        TT(M+2,M+2)=(CDABS(CI*RVK*CG(M+2,M+2)-1))**2

        DO I=2,M+1
        TT(1,I)=(CDABS(CG(1,I)))**2*RVK*RVI
        TT(I,I)=(CDABS(CI*RVI*CG(I,I)-1.))**2
        TT(I,M+2)=(CDABS(CG(I,M+2)))**2*RVK*RVI
        END DO


        DO I=2,M
        DO J=I+1,M+1
        TT(I,J)=(CDABS(CG(I,J)))**2*RVI**2
        END DO
        END DO

        DO I=M+1,2,-1
        DO J=1,I-1
        DO K=J,I-1
        TT(J,K)=TT(J,K)+TT(J,I)*TT(K,I)/(1.-TT(I,I))
        END DO
        END DO
        END DO

        TEF=(1.-TT(1,1))
        WRITE(10,*)EF,TEF
1000    CONTINUE

        CLOSE(10)
        STOP
        END
```

```
C Apéndice al Capitulo 4
~
C Programa para calcular la conductividad
~
C aplicando cadenas finitas en cada sitio de la muestra
~

~
       program CONBAR
~
c  ************************
~
C      Ef = Energia de Fermi investigada
~
C      N  = Numero de canales (sitios por capa)
C      Vx = Hoping entre capas
C      Vy = Hoping en la capa
C      cSL , cSR matrices complejas con las renormalizaciones
C             de los alambres
C      E_Hef  = Energia de Fermi menos el Hamiltoniano Efectivo
C      gi             = Localizador de capa renormalizado
C      cGLR   = Matriz de la Funcion de Green final
C      T(j,i) = Transmitancia entre canales i y j
c      totI   = Corriente total
c*****************************************************************

       IMPLICIT REAL*8 (E,F,O-Y)
        IMPLICIT COMPLEX*16 (A-d,G-H)
       IMPLICIT INTEGER (I-N)



       PARAMETER (rPI=3.1415926535897932D0)
       PARAMETER (N=1)       ! Eso calcula para solo dos sitios
       parameter(N2=N*2)
       character(60) filename, dir

        DIMENSION  cSR(N,N), cSL(N,N), Gi(N,N),cSwap(N,N)
     1    ,cVeff(N,N),ctVeff(N,N),rS(4)
     2  , cEHeff(N2,N2),cGLR(N2,N2),cSI(N,N)
     3  ,cEH(N,N),cG0L(N,N),cSk(N),rg(N)

       double precision RANDU
       complex*16 cS

       dir='c:\datos\'

       write(6,*) 'Ingrese nombre del archivo de salida:'
       read (5,*) filename
c      Energia de Fermi maxima y minima, cantidad de pasos, desorden,
c      largo de la muestra, acople a las cadenas, acople a los alambres, largo de las cadenas
       WRITE(6,*)'min max Ef?,NEnergias , DESORDEN?, larg? rVs? rVt? Lc?'
       READ(5,*)Emin,Emax,NE, W, Lm1, rVs, rVt, largoc
       Fc=0
      Einc=(Emax-Emin)/NE
       L=Lm1+1
       Np=N+1

       Vx=-1.D0
       Vy=-1.D0
        cIM=(0.D0,1.D0)
        cRE=(1.D0,0.D0)

c-----------------------------------------------------------------
c  -------------Comienza el laso  que varia parametros    -------

       DO 1000 Ef=Emin,Emax, Einc

        SEM1=.23821

        DO  I=1,N
         cEH(I,I)=(0.D0,0.D0)

           DO  J=I+1,N
          cEH(I,J)=(0.D0,0.D0)
          cEH(J,I)=(0.D0,0.D0)
          enddo
        enddo

c      Calculo la correccion debida a las cadenas
```

```
        do i=1,4
       rS(i)=1/Ef
        enddo

        do i=1,largoc
               rS(1)=1/(Ef-rS(1))
               rS(2)=1/(Ef-rS(2))
        enddo
c       Hago las cadenas de diferente largo
        rS(2)=1/(Ef-rS(2))

C*******************************************************
C-defino las autoenergias y los hoping efectivos (prcedimiento de decimation)
c        ________________________________
c         inicializo variables del Hamiltoniano efectivo
c        SR=0, SL=0, Veff=Vx*rUNIT

        do i=1,N
         do j=1,N
          cSR(i,j)=(0.d0,0.d0)
          cSL(i,j)=(0.d0,0.d0)
          cVeff(i,j)=(0.d0,0.d0)
         enddo
          cVeff(i,i)=Vx*cRE
        end do
C        --------------------------------
c       comienza eliminacion de (Lm-1) capas internas del sistema

          do it=1,Lm1

        cHOP=Vy*cdexp( 2.D0*cIM*rPI*Fc*it)
        cHOPc=dconjg(cHOP)

        do I=1,N
        Iu=I-1
         cEH(I,I)=(Ef-W*(RANDU(SEM1)-.5))*cRE
         If(I.GT.1) then
         cEH(I,Iu)=-cHOP
         cEH(Iu,I)=-cHOPC
        end if
         enddo !I

             call cSUM(Gi,1.d0,cEH,-1.d0,cSR,N)
             call cINV(Gi,N)

c            *****SL=SL+Veff(1/(E-Ei-SR))Veff*****
             call cPROD(cSwap,1.d0,cVeff,Gi,N)
                 call cTRAN(ctVeff,cVeff,N)
             call cPROD(cSR,1.d0,cSwap,cVeff,N) !SR temp. variable
             call cSUM(cSL,1.d0,cSL,1.d0,cSR,N)

c            ******SR=(V(i+1)**2)/(E-Ei-SR)*******
              call cEQUAL(cSR,Vx**2,Gi,N)

c            ******Veff=Veff*V(i+1)/(E-Ei-SR)******
             call cEQUAL(cVeff,Vx,cSwap,N)

          end do !it

c Defino las autoenergias de los alambres externos (en la base k)
        do k=1,N
        cSk(k)=(rVt/Vx)**2*cS( Ef-2.D0*Vy*dcos(k*rPI/Np) , Vx )
        end do

c      defino la matriz de las autoenergias de los alambres en la base de sitio

        do I1=1,N
         do I2=I1,N
         ctmp=(0.d0,0.d0)
          do k=1,N
        ctmp=ctmp+(2.D0/Np)*dsin(I1*k*rPI/Np)*dsin(I2*k*rPI/Np)*cSk(k)
          end do!k
         cSI(I1,I2)=ctmp
         cSI(I2,I1)=cSI(I1,I2)
        end do !I2
       end do !I1

C Defino el Hamiltoniano efectivo escrito en la base de la primera y
C ultima capa del sistema teniendo en cuenta los alambres
```

```
c     ___________Comienzo definiendo las contribuciones
C    del sistema y las decimaciones de los alambres
       do i=1,N
          do j=1,N
           cEHeff(i,j)=    -cSL(i,j)-cSI(i,j)
           cEHeff(N+i,N+j)=-cSR(i,j)-cSI(i,j)
           cEHeff(i,N+j)=   -cVeff(i,j)
           cEHeff(j+N,i)= -dconjg(cVeff(i,j))  !cEHeff(i,j+N)! right because real
          end do !j
c     Agrego las energias diagonales no perturbadas de la capa
c      Con estas sumo la accion de las cadenas laterales
       cEHeff(i,i)=cEHeff(i,i)+(Ef-W*(RANDU(SEMI)-.5))*cRE
       1-(rVs/Vx)**2*rS(1)

      cEHeff(N+i,N+i)=cEHeff(N+i,N+i)+(Ef-W*(RANDU(SEMI)-.5))*cRE
       1-(rVs/Vx)**2*rS(2)

       end do ! i

c     Agrego los hopings en la capa
       do i=1,N-1
          cEHeff(i,i+1)=cEHeff(i,i+1)-Vy
         cEHeff(i+1,i)=cEHeff(i+1,i)-Vy
         cEHeff(N+i,N+i+1)=cEHeff(N+i,N+i+1)
     *           -Vy*cdexp(-2.D0*cIM*rPI*Fc*L)
         cEHeff(N+i+1,N+i)=cEHeff(N+i+1,N+i)
     *           -Vy*cdexp( 2.D0*cIM*rPI*Fc*L)
       end do   !

       call cEQUAL(cGLR,1.d0,cEHeff,N2)


       call cINV(cGLR,N2)



c ***********************************************************



c transformo la funcion de Green a la base k
       do k1=1,N
       do k2=1,N
       ct=(0.D0,0.D0)
         do I1=1,N
         do I2=1,N
        ct=ct+(2.D0/Np)*dsin(I1*k1*rPI/Np)*cGLR(I1,N+I2)*
     *    dsin(I2*k2*rPI/Np)
        end do
        end do
       cG0L(k1,k2)=ct
       end do
       enddo

c***Calculo de la conductancia

       rgt=0.D0
       Do k1=1,N
       rg(k1)=0.d0
       do k2=1,N
       rg(k1)=rg(k1)+
     1  4.D0*dimag(cSk(k1))*cdabs(cG0L(k1,k2))**2*dimag(cSk(k2))
       end do  !k2
       rgt=rgt+rg(k1)
       end do !k1

       open (11,FILE=trim(dir)//trim(filename)//'.dat',STATUS='UNKNOWN'
       1   ,POSITION='APPEND')
c      Salida: Energia, conductancia total
        IF(N.GE.1) WRITE(11,*) Ef, rgt
       close(11)


1000   CONTINUE ! va a buscar un nuevo parametro
        END

C-----------------------------------------------------------------
C-----------------------------------------------------------------
C-----------------------------------------------------------------
```

```
      DOUBLE PRECISION FUNCTION RANDU(DSEED)
      DOUBLE PRECISION DSEED
      DOUBLE PRECISION D2P31M, D2P31
      DATA  D2P31M/2147483647.D0/,D2P31/2147483649.D0/
      DSEED=DMOD(16807.D0*DSEED,D2P31M)
      RANDU=DSEED/D2P31
      RETURN
      END

      complex*16 function cS(E,V)
      real*8 E,rdisc,V
      complex*16 cRE, cIM
      cRE=(1.D0,0.D0)
      cIM=(0.D0,1.D0)
      rdisc=(E/2.)**2-V**2
      if(rdisc.gt.0.) then
       cS=( E/2.-(E/dabs(E))*dsqrt(rdisc))*cRE
      else
      cS=E/2.*cRE-cIM*dsqrt(-rdisc)
      endif
      return
      end


      subroutine cINV(cA,N)
      PARAMETER (NMAX=500)
      IMPLICIT COMPLEX*16 (C)
      implicit real *8(a-b,d-h,o-z)
      DIMENSION cA(N,N),IPIV(NMAX),INDXR(NMAX),INDXC(NMAX)
       cRE=(1.,0.)

      DO 11 J=1,N
        IPIV(J)=0
11    CONTINUE
      DO 22 I=1,N
        BIG=0.
        DO 13 J=1,N
          IF(IPIV(J).NE.1)THEN
            DO 12 K=1,N
              IF (IPIV(K).EQ.0) THEN
                IF (cdABS(cA(J,K)).GE.BIG)THEN
                  BIG=cdABS(cA(J,K))
                  IROW=J
                  ICOL=K
                ENDIF
              ELSE IF (IPIV(K).GT.1) THEN
                PAUSE 'Singular matrix'
              ENDIF
12          CONTINUE
          ENDIF
13      CONTINUE
        IPIV(ICOL)=IPIV(ICOL)+1
        IF (IROW.NE.ICOL) THEN
          DO 14 L=1,N
            cDUM=cA(IROW,L)
            cA(IROW,L)=cA(ICOL,L)
            cA(ICOL,L)=cDUM
14        CONTINUE
         ENDIF
        INDXR(I)=IROW
        INDXC(I)=ICOL
        IF (cdabs(cA(ICOL,ICOL)).lt.1.d-107) PAUSE 'Singular matrix.'
        cPIVINV=cRE/cA(ICOL,ICOL)
        cA(ICOL,ICOL)=cRE
        DO 16 L=1,N
          cA(ICOL,L)=cA(ICOL,L)*cPIVINV
16      CONTINUE
        DO 21 LL=1,N
          IF(LL.NE.ICOL)THEN
            cDUM=cA(LL,ICOL)
            cA(LL,ICOL)=0.*cRE
            DO 18 L=1,N
              cA(LL,L)=cA(LL,L)-cA(ICOL,L)*cDUM
18          CONTINUE

          ENDIF
21      CONTINUE
22    CONTINUE
      DO 24 L=N,1,-1
        IF(INDXR(L).NE.INDXC(L))THEN
```

```
          DO 23 K=1,N
            cDUM=cA(K,INDXR(L))
            cA(K,INDXR(L))=cA(K,INDXC(L))
            cA(K,INDXC(L))=cDUM
23        CONTINUE
        ENDIF
24    CONTINUE
      RETURN
      END

c*****************************************************************

       subroutine cPROD(cD,r1,cA,cB,n)
C       product of two square matrices
C       D= r1* A* B
       complex*16 cA(n,n),cB(n,n),cD(n,n)
       real*8 r1
       do 3 i=1,n
         do 2 j=1,n
         cD(i,j)=(0.d0,0.d0)
         do 1 k=1,n
  1       cD(i,j)= cD(i,j)+r1*cA(i,k)*cB(k,j)
  2    continue
  3    continue
       return
       end


       subroutine cSUM(D,a1,A,a2,B,n)
C       evaluates the sum of two matrices
C       a1*A + a2* B= D
       complex*16 A(n,n),B(n,n), D(n,n)
       real*8 a1,a2
       do 2 i=1,n
         do 1 j=1,n
  1       D(i,j)=a1*A(i,j)+a2*B(i,j)
  2    continue
       return
       end



       subroutine cEQUAL(cD, r1,cA,n)
c       evaluates cD= r1* cA
       complex*16 cA(n,n),cD(n,n)
       real*8 r1
       do 2 i=1,n
         do 1 j=1,n
   1     cD(i,j)= r1* cA(i,j)
   2   continue
       return
       end


       subroutine cTRAN(ctV,cV,N)
c         evaluates the transpose conjugate
        complex*16 ctV(N,N),cV(N,N)
        do i=1,N
        do j=1,N
        ctV(i,j)=dconjg(cV(j,i))
       end do
       enddo
       return
       end
```