```
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! This code does the following:
!
!   1) Diagonalize Hi and Hf
!   Write the Hamiltonians in the SITE-BASIS and diagonalize them.
!   VecF = eigenstates of the FINAL Hamiltonian in the site-basis
!   EigF = eigenvalues of the FINAL Hamiltonian
!   VecI = eigenstates of the INITIAL Hamiltonian in the site-basis
!   EigI = eigenvalues of the INITIAL Hamiltonian
!
!   2) The initial state is one of the VecI's. It is selected according to a
!   chosen T (temperature) so that Eini ~ E_T (E_T is obtained from the
canonical distribution)
!
!   3) Project the initial state onto the eigenstates of the FINAL
Hamiltonian and evolve it.
!   The eigenvalues of the FINAL Hamiltonian and the projections of the
initial state on
!   these final eigenstates are given in an output file.
!
!   4) Time evolution of the FIDELITY
!   The results are given in an output file.
!
!
!
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!             VARIABLES to be used in the whole code
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       module variables

       implicit none

       integer (kind=4) :: i,j,k,ii,jj,kk,ij,ik,jk
       integer (kind=4) :: chain_size, upspins
       integer (kind=4) :: dimTotal,dd

! PARAMETERS of the FINAL Hamiltonian
       real (kind=8) :: JxyF,JzF,defectF,alphaF,defborder

! PARAMETERS of the INITIAL Hamiltonian
       real (kind=8) :: JxyI,JzI,defectI,alphaI

! SITE-BASIS
```

```
      integer (kind=4), dimension(:,:), allocatable :: site_basis

! Eigenvectors and eigenvalues of the FINAL Hamiltonian
      real (kind=8), dimension(:,:), allocatable :: VecF
      real (kind=8), dimension(:), allocatable :: EigF
! Eigenvectors and eigenvalues of the INITIAL Hamiltonian
      real (kind=8), dimension(:,:), allocatable :: VecI
      real (kind=8), dimension(:), allocatable :: EigI

! for INITIAL STATE
      INTEGER (kind=4) :: este
      real (kind=8) :: zero,menor,Zpartition,Eaux,tempera
! Eigenvectors of Hi projected onto states of Hf
      real (kind=8), dimension(:,:), allocatable :: VecImf
! To search for the desired initial state
      real (kind=8) :: EigImf,EigImfsq,InitialEne

! for the DIAGONALIZATION
      INTEGER (kind=4) :: INFO
      real (kind=8), dimension(:), allocatable :: work


! TIME EVOLUTION
      real (kind=8), dimension(:), allocatable :: Calpha
      INTEGER (kind=4) :: tt,tinitial,tfinal
      real (kind=8) :: dt,time,auxCos,auxSin
      real (kind=8) :: FIDEL

! For the OUTPUT files
      character(len=2) LC
      character(len=3) inC
      character(len=4) dF,zF
      character(len=4) dI,zI,Ai,dc
      character(len=5) Af
      character(len=70) saiS,saiIn

      end module



!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!             Program starts here
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      Program Fidelity

      use variables
```

```
      implicit none


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! PARAMETERS
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      chain_size=18
      upspins=6
      dimTotal=18564

!      chain_size=16
!      upspins=8
!      dimTotal=12870

!      chain_size=15
!      upspins=5
!      dimTotal=3003

!      chain_size=12
!      upspins=6
!      dimTotal=924


!      chain_size=6
!      upspins=3
!      dimTotal=20

      dd=dimTotal

      JxyI=1.0d0
      JxyF=1.0d0
      defborder=0.0d0


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Parameters of Hi and Hf
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      JzI=0.5d0
      defectI=0.0d0
      alphaI=0.0d0
      JzF=0.5d0
      defectF=0.0d0
      alphaF=1.0d0
      tempera=4.4d0
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```fortran
! TIME EVOLUTION
       tinitial=0
       tfinal=1000
       dt=0.01d0
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!




!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! OUTPUT FILES
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      write(LC,8) chain_size
8    format(i2.2)
      write(dI,11) defectI
      write(zI,11) JzI
      write(Ai,11) alphaI
      write(dF,11) defectF
      write(zF,11) JzF
      write(dc,11) defborder
11   format(F4.2)
       write(Af,12) alphaF
12   format(F5.3)

! LDOS   cccccccccccccccccccccccccccccccccccc

saiS='Calpha_L'//LC//'Zi'//zI//'Ai'//Ai//'_Zf'//zF//'Af'//Af//'db'//dc//'.dat'
       OPEN(unit=20, FILE=saiS,STATUS='UNKNOWN')
! Fid     cccccccccccccccccccccccccccccccc

saiIn='Fid_L'//LC//'Zi'//zI//'Ai'//Ai//'_Zf'//zF//'Af'//Af//'db'//dc//'.dat'
       OPEN(unit=30, FILE=saiIn,STATUS='UNKNOWN')



!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c CREATE SITE-BASIS and defining PARITY of the basis
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       allocate(site_basis(dimTotal,chain_size))
       call SiteBasis()


! ALLOCATION needed during the whole CODE

       allocate(EigI(dimTotal))
       allocate(EigF(dimTotal))
```

```
      allocate(VecI(dimTotal,dimTotal))
      allocate(VecF(dimTotal,dimTotal))

      allocate(work(7*dimTotal))

      allocate(Calpha(dimTotal))



!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c INITIAL HAMILTONIAN in the SITE-BASIS
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      call HamiltonianINITIAL()
      CALL DSYEV('V','U',dimTotal,VecI,dimTotal,EigI,WORK,7*dimTotal,INFO)

!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c FINAL HAMILTONIAN in the SITE-BASIS
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      call HamiltonianFINAL()
      CALL DSYEV('V','U',dimTotal,VecF,dimTotal,EigF,WORK,7*dimTotal,INFO)



!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c EIGENSTATES VecI projected onto states of the FINAL Hamiltonian
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!
! with a library DGEMM to multiply matrices
!
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      allocate(VecImf(dimTotal,dimTotal))
      call DGEMM('t','n',dd,dd,dd,1.0d0,VecF,dd,VecI,dd,0.0d0,VecImf,dd)


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! Below ou can choose if to select the initial state with Eini~0
! or
! an initial state with Eini~E_T
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! Find the |ini> with <ini|Hf|ini> closest to zero
!       zero = -1000000.0d0
!       DO i=1,dimTotal
!       EigImf=0.0d0
```

```
!        Do j=1,dimTotal
!        EigImf=EigImf+EigF(j)*(VecImf(j,i)**2)
!        Enddo
!        If(EigImf.lt.0.0.and.EigImf.ge.zero) then
!          este=i
!          zero=EigImf
!        Endif
!       ENDDO
!       write(20,105) '#','which state is closest to E=0:',este
!       write(20,110) '#','E_alpha','Calpha'
!
!        Do j=1,dimTotal
!         Calpha(j)=VecImf(j,este)
!        write(20,120) EigF(j),Calpha(j)
!        ENDDO
!       deallocate(VecImf)
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! Find the |ini> with <ini|Hf|ini> ~ E_T
       write(20,106) '#','Temperature=',tempera
       Eaux=0.0d0
       Zpartition=0.0d0
       Do i=1,dimTotal
        Eaux=Eaux+EigF(i)*exp(-EigF(i)/tempera)
        Zpartition=Zpartition+exp(-EigF(i)/tempera)
       Enddo
       Eaux=Eaux/Zpartition
       write(20,106) '#','Ecanonical=',Eaux

       menor=10000.0d0
       Do i=1,dimTotal
       EigImf=0.0d0
        Do j=1,dimTotal
        EigImf=EigImf+EigF(j)*(VecImf(j,i)**2)
        Enddo
        If(dabs(EigImf - Eaux).lt.menor) then
          menor=dabs(EigImf - Eaux)
          InitialEne=EigImf
          este=i
        Endif
       Enddo
       write(20,106) '#','Closest initial energy is',InitialEne
       write(20,110) '#','E_alpha','Calpha'
        Do j=1,dimTotal
         Calpha(j)=VecImf(j,este)
        write(20,120) EigF(j),Calpha(j)
        ENDDO
       deallocate(VecImf)
```

```
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! TIME DEPENDENCE of the INITIAL STATE
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       DO tt=tinitial,tfinal
         time=dble(tt)*dt
         auxCos=0.0d0
         auxSin=0.0d0
         Do i=1,dimTotal
           auxCos=auxCos+( Calpha(i)**2 )*dcos( time*EigF(i) )
           auxSin=auxSin+( Calpha(i)**2 )*dsin( time*EigF(i) )
         Enddo
! FIDELITY inside the TIME-LOOP
         FIDEL=auxCos**2 + auxSin**2
         write(30,130) time, FIDEL
! end the TIME-LOOP
       ENDDO


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! FORMAT
105    format(A1,2X,A30,I6)
106    format(A1,2X,A30,E18.9)
110    format(A1,3X,A8,8X,A8)
120    format(3E18.9)
130    format(F8.2,E18.9)
        close(20)
        close(30)

! GET SPACE by DEALLOCATION
       deallocate(site_basis)

       deallocate(EigI)
       deallocate(EigF)

       deallocate(VecI)
       deallocate(VecF)

       deallocate(work)

       deallocate(Calpha)
!cccccccccccccccccccccccccccccccccccccccccccccccccccc
!c END END END END END END END END END END END END
!cccccccccccccccccccccccccccccccccccccccccccccccccccc
!      STOP
       END
```

```
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccc SUBROUTINES SUBROUTINES SUBROUTINES cccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! *************** WRITING THE SITE-BASIS **************************
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SiteBasis()
      use variables
      implicit none

      INTEGER (kind=4) :: ib,jb

      logical mtc
      integer (kind=4) :: in(chain_size),m2,h

      mtc=.false.

!c INITIALIZATION
      Do ib=1,dimTotal
        Do jb=1,chain_size
          site_basis(ib,jb)=0
        enddo
      enddo

      ii=1

71   call nexksb(chain_size,upspins,in,mtc,m2,h)
      Do jb=1,upspins
        site_basis(ii,in(jb))=1
      Enddo
      ii=ii+1
      if(mtc) goto 71

!c END of SUBROUTINE for SITE-BASIS
      return
      end subroutine SiteBasis
!ccccccccccccccccccccccccccccccccccccccccc
!c SUBROUTINE to get the COMBINATIONS
!ccccccccccccccccccccccccccccccccccccccccc
      subroutine nexksb(n,k,a,mtc,m2,h)
      integer (kind=4) :: n,k,a(n),m2,h,jn
      logical mtc
```

```
      if(.not.mtc) then
      m2=0
      h=k
      go to 50
      endif
      if(m2.lt.n-h) h=0
      h=h+1
      m2=a(k+1-h)
   50 do jn=1,h
      a(k+jn-h)=m2+jn
      enddo
      mtc=a(1).ne.n-k+1

      return
      end subroutine nexksb
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!
! SUBROUTINE to write the INITIAL HAMILTONIAN in the SITE-BASIS
!
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       subroutine HamiltonianINITIAL()

       use variables
       implicit none

       INTEGER (kind=4) :: tot,bIp(chain_size)


!c INITIALIZATION
       Do i=1,dimTotal
         Do j=1,dimTotal
           VecI(i,j)=0.0d0
         Enddo
       Enddo
```

```
! DIAGONAL ELEMENTS
*********************************************************************

       Do i=1,dimTotal

!cccccccc DEFECT on SITE 1 ccccccccccccccc
         VecI(i,i)=VecI(i,i)+0.5d0*defborder*(-1.0d0)**(1+site_basis(i,1))

!cccccccc DEFECT on MIDDLE ccccccccccccccc
         VecI(i,i)=VecI(i,i)+0.5d0*defectI*(-1.0d0)**
(1+site_basis(i,chain_size/2))

!ccccccccccccccc NN cccccccccccccccccccccc
         Do j=1,chain_size-1
       VecI(i,i)=VecI(i,i)+(JzI/4.d0)*(-1.0d0)**
(site_basis(i,j)+site_basis(i,j+1))
         enddo

!ccccccccccccccc NNN cccccccccccccccccccccc
         Do j=1,chain_size-2
       VecI(i,i)=VecI(i,i)+alphaI*(JzI/4.d0)*(-1.0d0)**
(site_basis(i,j)+site_basis(i,j+2))
         enddo

! CLOSING i=1,dimTotal
       enddo
! END of DIAGONAL
**********************************************************************




! OFF-DIAGONAL ELEMENTS
*******************************************************************
       Do i = 1, dimTotal-1
         Do j = i+1, dimTotal

           tot = 0
           Do k = 1, chain_size
             bip(k) = mod(site_basis(i,k) + site_basis(j,k),2)
             tot = tot + bip(k)
           enddo

           IF(tot.EQ.2) then

!ccccccccccccccc NN cccccccccccccccccccccc
```

```
            do k = 1, chain_size-1
              IF(bip(k)*bip(k+1).EQ.1) then
                VecI(i,j)=VecI(i,j)+JxyI/2.0d0
                VecI(j,i)=VecI(j,i)+JxyI/2.0d0
              ENDIF
            enddo

!cccccccccccccc NNN cccccccccccccccccccc
            do k = 1, chain_size-2
              IF(bip(k)*bip(k+2).EQ.1) then
                VecI(i,j)=VecI(i,j)+alphaI*JxyI/2.0d0
                VecI(j,i)=VecI(j,i)+alphaI*JxyI/2.0d0
              ENDIF
            enddo

! CLOSING IF for tot=2
            ENDIF
!c CLOSING Do i=1,dimTotal-1 and Do j=i+1,dimTotal
          enddo
        enddo


!  END of SUBROUTINE that constructs the INITIAL HAMILTONIAN
!  in the SITE-BASIS
        return
        end subroutine HamiltonianINITIAL
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc




!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!
! SUBROUTINE to write the FINAL HAMILTONIAN in the SITE-BASIS
!
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine HamiltonianFINAL()

      use variables
      implicit none

      INTEGER (kind=4) :: tot,DifferentSite(chain_size)
```

```
!c INITIALIZATION
      Do i=1,dimTotal
        Do j=1,dimTotal
          VecF(i,j)=0.0d0
        Enddo
      Enddo


! DIAGONAL ELEMENTS
**********************************************************************
      Do i=1,dimTotal
!cccccccc DEFECT on SITE 1 ccccccccccccccc
        VecF(i,i)=VecF(i,i)+0.5d0*defborder*(-1.0d0)**(1+site_basis(i,1))
!cccccccc DEFECT on MIDDLE ccccccccccccccc
        VecF(i,i)=VecF(i,i)+0.5d0*defectF*(-1.0d0)**
(1+site_basis(i,chain_size/2))
!ccccccccccccccc NN cccccccccccccccccccccc
        Do j=1,chain_size-1
          VecF(i,i)=VecF(i,i)+(JzF/4.d0)*(-1.0d0)**
(site_basis(i,j)+site_basis(i,j+1))
        enddo
!ccccccccccccccc NNN cccccccccccccccccccc
         Do j=1,chain_size-2
       VecF(i,i)=VecF(i,i)+alphaF*(JzF/4.d0)*(-1.0d0)**
(site_basis(i,j)+site_basis(i,j+2))
         enddo
! CLOSING i=1,dimTotal
      enddo
! END of DIAGONAL
***********************************************************************




! OFF-DIAGONAL ELEMENTS
*******************************************************************
      Do i = 1, dimTotal-1
        Do j = i+1, dimTotal

          tot = 0
          Do k = 1, chain_size
            DifferentSite(k)=0
          Enddo

          Do k = 1, chain_size
            If( site_basis(i,k).ne.site_basis(j,k) ) then
```

```
            tot = tot + 1
            DifferentSite(tot)=k
            Endif
          Enddo

          IF(tot.EQ.2) then
!cccccccccccccccc NN cccccccccccccccccccccccccccccccc
          IF((DifferentSite(2) - DifferentSite(1)).EQ.1) then
            VecF(i,j)=VecF(i,j)+JxyF/2.0d0
            VecF(j,i)=VecF(i,j)
          Endif
!cccccccccccccccc NNN cccccccccccccccccccccccccccccccc
         IF((DifferentSite(2) - DifferentSite(1)).EQ.2) then
           VecF(i,j)=VecF(i,j)+alphaF*JxyF/2.0d0
           VecF(j,i)=VecF(i,j)
          ENDIF
! CLOSING IF for tot=2
          ENDIF

!c CLOSING Do i=1,dimTotal-1 and Do j=i+1,dimTotal
        enddo
      enddo


!  END of SUBROUTINE that constructs the HAMILTONIAN
!  in the SITE-BASIS
      return
      end subroutine HamiltonianFINAL
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```