```
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! This code does the following:
!
!   *) Write the total INITIAL Hamiltonian in the SITE-BASIS and diagonalize it to find the
INITIAL state
!
!   *) Write the total FINAL Hamiltonian in the SITE-BASIS and diagonalize it
!
!   *) Write the INITIAL state in terms of the eigenstates of the FINAL Hamiltonian. Use DGEMV.
!       Select the state according to its Eini and desired temperature.
!       Use the components of the projection to compute the observables in the diagonal
ensemble.
!
!   *) Compute all eigenstate expectation values of various observables.
!       Use them to compute the results from the observables from the
!                                       microcanonical and diagonal ensembles
!
!   *) With this code you can solve EXERCISE 13, but the spin current is not provided.
!      You can solve EXERCISE 14.
!      You can solve EXERCISE 15 and get Figs.11 and 12. You will need to add a piece to
!      compute the extremal fluctuations.
!      You can solve EXERCISE 16 by getting the results for the diagonal ensemble and
!      microcanonical ensemble. You will need to add a loop for the perturbations to get Fig.13
!      and loop for temperature to get Fig.14.
!
!
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

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


       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


! FIXED PARAMETERS
       real (kind=8) :: Jxy,defborder

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

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

! SITE-BASIS
       integer (kind=4), dimension(:,:), allocatable :: site_basis

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

```
! Eigenvectors and eigenvalues for the INITIAL and FINAL Hamiltonians
      real (kind=8), dimension(:,:), allocatable :: VecI,VecF
      real (kind=8), dimension(:,:), allocatable :: HamMF,Haux
      real (kind=8), dimension(:), allocatable :: EigI,EigF

! For the DELOCALIZATION MEASURES
      real (kind=8) :: IPR,Shan,auxIPR

! INITIAL state and its TEMPERATURE and initial ENERGY
      real (kind=8), dimension(:), allocatable :: Initial,auxIni
      real (kind=8) :: Temperature, Etemp, Einitial, auxEn, EfromHI
      real (kind=8) :: aux, partition, menor,width
      integer (kind=4) :: qual

! for the OBSERVABLES
      real (kind=8) :: IE,KE
      real (kind=8) :: Czz,Czz2,Czz3
      real (kind=8) :: Cxx,Cxx2,Cxx3
      integer (kind=4) :: nxx,nxx2,nxx3
      real (kind=8), dimension(:), allocatable :: IEdiag,SiteMag
      real (kind=8), dimension(:), allocatable :: Czzaux,Czza2,Czza3
      real (kind=8), dimension(:), allocatable :: ZZ
      real (kind=8), dimension(:), allocatable :: Stzz,Stxx
      real (kind=8) :: phase,phaseMag
      real (kind=8) :: auxZ,auxX
      INTEGER (kind=4), dimension(:), allocatable :: Npair,cxxI,cxxJ
      INTEGER (kind=4), dimension(:), allocatable :: cxxI2,cxxJ2,cxxI3,cxxJ3
      INTEGER (kind=4), dimension(:,:), allocatable :: pairI,pairJ
      real (kind=8) :: mMI !how many states in micro window (real number)

! For the DIAGONAL ENSEMBLE
      real (kind=8), dimension(:), allocatable :: Calpha,CalphaAUX
      real (kind=8) :: IEde,KEde
      real (kind=8) :: Czzde,Czz2de,Czz3de
      real (kind=8) :: Cxxde,Cxx2de,Cxx3de
      real (kind=8), dimension(:), allocatable :: SiteMagDE
      real (kind=8), dimension(:), allocatable :: StzzDE,StxxDE

! For the MICROCANONICAL ENSEMBLE
      integer (kind=4) :: numMIC
      real (kind=8) :: delta,deltaMIC
      real (kind=8) :: IEmic,KEmic
      real (kind=8) :: Czzmic,Czz2mic,Czz3mic
      real (kind=8) :: Cxxmic,Cxx2mic,Cxx3mic
      real (kind=8), dimension(:), allocatable :: SiteMagMIC
      real (kind=8), dimension(:), allocatable :: StzzMIC,StxxMIC

! CONSTANTS
      real (kind=8) :: pi

! For the OUTPUT files
      character(len=70) saiCalp,saiIPR,saiINITIAL
      character(len=70) saiIEKE,saiCz,saiCx,saiStz,saiStx
      character(len=70) saiAllKE, saiAllStx,saiAllCx
      character(len=70) saiAllStz,saiAllCz

      end module
```

```
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!             Program starts here
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       Program Model

       use variables
       implicit none

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! PARAMETERS
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       chain_size=18
       upspins=chain_size/3
       dimTotal=18564

!       chain_size=15
!       upspins=chain_size/3
!       dimTotal=3003

!       chain_size=12
!       upspins=chain_size/3
!       dimTotal=495

!       chain_size=9
!       upspins=chain_size/3
!       dimTotal=84

!       chain_size=6
!       upspins=chain_size/3
!       dimTotal=15

       dd=dimTotal

! Fixed parameters
       Jxy=1.0d0
       defborder=0.1d0

!ccccc CONSTANT PI ccccccccccc
       pi=dacos(-1.0d0)
!ccccccccccccccccccccccccccccc



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!        CHOOSE VALUES
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! Temperature of the initial state
       Temperature =7.0d0

! Initial Hamiltonian: XXZ
       JzI=0.48d0
       alphaI=0.0d0
       defectI=0.0d0
! Quench to NNN model
       JzF=0.48d0
```

```
       alphaF=0.44d0
       defectF=0.0d0
! Quench to impurity model
       JzF=0.48d0
       alphaF=0.0d0
       defectF=0.9d0

! For the MICROCANONICAL ensemble
       deltaMIC=0.4d0
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!





!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!                         OUTPUT FILES
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

! COMPONENTS   cccccccccccccccccccccc
       saiCalp='CalphaL18Ji0.48Jf0.48ai0.00af0.44dbord0.1defF0.00T7.dat'
       OPEN(unit=16, FILE=saiCalp,STATUS='UNKNOWN')

! IPR   cccccccccccccccccccccc
       saiIPR='IPR_L18Ji0.48Jf0.48ai0.00af0.44dbord0.1defF0.00.dat'
       OPEN(unit=17, FILE=saiIPR,STATUS='UNKNOWN')

!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!                 DIAGONAL and MICROCANONICAL ENSEMBLES
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc!
! IE and KE for DE and ME  cccccccccccccccccccccc
       saiIEKE='DEME_IEKE_L18Ji0.48Jf0.48ai0.00af0.44dbord0.1defF0.00.dat'
       OPEN(unit=20, FILE=saiIEKE,STATUS='UNKNOWN')

! Cz for DE and ME  cccccccccccccccccccccc
       saiCz='DEMECz_L18Ji0.48Jf0.48ai0.00af0.44dbord0.1defF0.00.dat'
       OPEN(unit=22, FILE=saiCz,STATUS='UNKNOWN')

! Cx for DE and ME  cccccccccccccccccccccc
       saiCx='DEMECx_L18Ji0.48Jf0.48ai0.00af0.44dbord0.1defF0.00.dat'
       OPEN(unit=24, FILE=saiCx,STATUS='UNKNOWN')

! Stz for DE and ME  cccccccccccccccccccccc
       saiStz='DEMEStz_L18Ji0.48Jf0.48ai0.00af0.44dbord0.1defF0.00.dat'
       OPEN(unit=26, FILE=saiStz,STATUS='UNKNOWN')

! Stx for DE and ME  cccccccccccccccccccccc
       saiStx='DEMEStx_L18Ji0.48Jf0.48ai0.00af0.44dbord0.1defF0.00.dat'
       OPEN(unit=28, FILE=saiStx,STATUS='UNKNOWN')

!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!                  EIGENSTATE EXPECTATION VALUES
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! INTERACTION and KINETIC energies for HamFINAL  cccccccccccccccccccccc
       saiAllKE='EEV_KEIEL18Ji0.48Jf0.48ai0.00af0.44dbord0.1defF0.00.dat'
       OPEN(unit=30, FILE=saiAllKE,STATUS='UNKNOWN')

!Cz for HamFINAL  cccccccccccccccccccccc
       saiAllCz='EEV_CzL18Ji0.48Jf0.48ai0.00af0.44dbord0.1defF0.00.dat'
```

```
       OPEN(unit=32, FILE=saiAllCz,STATUS='UNKNOWN')

!Cx for HamFINAL  ccccccccccccccccccccc
       saiAllCx='EEV_CxL18Ji0.48Jf0.48ai0.00af0.44dbord0.1defF0.00.dat'
       OPEN(unit=34, FILE=saiAllCx,STATUS='UNKNOWN')

! Stzz for HamFINAL  ccccccccccccccccccccc
       saiAllStz='EEV_StzL18Ji0.48Jf0.48ai0.00af0.44dbord0.1defF0.00.dat'
       OPEN(unit=36, FILE=saiAllStz,STATUS='UNKNOWN')

! Stxx for HamFINAL  ccccccccccccccccccccc
       saiAllStx='EEV_StxL18Ji0.48Jf0.48ai0.00af0.44dbord0.1defF0.00.dat'
       OPEN(unit=38, FILE=saiAllStx,STATUS='UNKNOWN')

! INITIAL state  ccccccccccccccccccccc
       saiINITIAL='Initial.L18Ji0.48Jf0.48ai0.00af0.44dbord0.1defF0.00.dat'
       OPEN(unit=40, FILE=saiINITIAL,STATUS='UNKNOWN')



!ccccccccccccccccccccccccc ALLOCATION ccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! ALLOCATION for the vectors and energies
       allocate(site_basis(dimTotal,chain_size))
       allocate(Initial(dimTotal))
       allocate(auxIni(dimTotal))
       allocate(VecF(dimTotal,dimTotal))
       allocate(EigF(dimTotal))

! ALLOCATION for the DIAGONAL ENSEMBLE
       allocate(Calpha(dimTotal))
       allocate(SiteMagDE(chain_size))
       allocate(StzzDE(0:chain_size/2))
       allocate(StxxDE(0:chain_size/2))

! ALLOCATION for the MICROCANONICAL ENSEMBLE
       allocate(SiteMagMIC(chain_size))
       allocate(StzzMIC(0:chain_size/2))
       allocate(StxxMIC(0:chain_size/2))

! ALLOCATION for the observables

       allocate(IEdiag(dimTotal))
       allocate(SiteMag(chain_size))

       allocate(Czzaux(dimTotal))
       allocate(Czza2(dimTotal))
       allocate(Czza3(dimTotal))

       allocate(cxxI(dimTotal/2))
       allocate(cxxJ(dimTotal/2))
       allocate(cxxI2(dimTotal/2))
       allocate(cxxJ2(dimTotal/2))
       allocate(cxxI3(dimTotal/2))
       allocate(cxxJ3(dimTotal/2))

       allocate(ZZ(0:chain_size/2))
       allocate(Stzz(0:chain_size/2))
```

```
       allocate(Npair(chain_size-1))
       allocate(pairI(chain_size-1,(chain_size-1)*(dimTotal-1)))
       allocate(pairJ(chain_size-1,(chain_size-1)*(dimTotal-1)))
       allocate(Stxx(0:chain_size/2))
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c CREATE SITE-BASIS
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       call SiteBasis()
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c EIGENVALUES and EIGENSTATES in the SITE-BASIS of the
!                                                 INITIAL Hamiltonian
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       allocate(VecI(dimTotal,dimTotal))
       allocate(EigI(dimTotal))
       call HamiltonianINITIAL()
       allocate(work(7*dimTotal))
       CALL DSYEV('V','U',dimTotal,VecI,dimTotal,EigI,WORK,7*dimTotal,INFO)
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!               IPR and Shannon Entropy in the XXZ basis
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! Before diagonalizing the FINAL Hamiltonian in the site basis, VecF
! we can use it to get the equivalent matrix in the XXZ-basis: HamMF
! which, after diagonalization, will give the eigenstates in the XXZ-basis
! XXZ-basis = Mean-field-basis = eigenstates of Hinitial
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       call HamiltonianFINAL()
       call IPRShannon()
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!                  FINAL EIGENSTATES in the SITE basis
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       CALL DSYEV('V','U',dimTotal,VecF,dimTotal,EigF,WORK,7*dimTotal,INFO)
       deallocate(work)
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!                               INITIAL state
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       allocate(CalphaAUX(dimTotal))
       call InitialStateXXZ()
! deallocate
       deallocate(VecI)
       deallocate(EigI)
       deallocate(CalphaAUX)

! ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! Output for Calphas
       write(16,190) '#','Energy','Calpha'
       Do k=1,dimTotal
         write(16,116) EigF(k), Calpha(k)
       Enddo
190    format(A1,1X,A13,3X,A13)
116    format(2E18.9)
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc




!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!             OBSERVABLES in the DIAGONAL and MICROCANONICAL ensembles
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       call Observables()

       close(16)
       close(17)
       close(20)
       close(30)
       close(32)
       close(34)
       close(36)
       close(38)
       close(40)


! GET SPACE by DEALLOCATION
       deallocate(site_basis)

       deallocate(Initial)
       deallocate(VecF)
       deallocate(EigF)

       deallocate(Calpha)
       deallocate(SiteMagDE)
       deallocate(StzzDE)
       deallocate(StxxDE)

       deallocate(SiteMagMIC)
       deallocate(StzzMIC)
       deallocate(StxxMIC)

       deallocate(IEdiag)
       deallocate(SiteMag)
```

```
        deallocate(Czzaux)
        deallocate(Czza2)
        deallocate(Czza3)

        deallocate(cxxI)
        deallocate(cxxJ)
        deallocate(cxxI2)
        deallocate(cxxJ2)
        deallocate(cxxI3)
        deallocate(cxxJ3)

        deallocate(ZZ)
        deallocate(Stzz)

        deallocate(Npair)
        deallocate(pairI)
        deallocate(pairJ)
        deallocate(Stxx)

!ccccccccccccccccccccccccccccccccccccccccccccccccc
!c END END END END END END END END END END END END
!ccccccccccccccccccccccccccccccccccccccccccccccccc
!       STOP
        END



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccc SUBROUTINES SUBROUTINES SUBROUTINES ccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! *************** WRITING THE SITE-BASIS ***************************
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!
!       SUBROUTINE to write the HAMILTONIAN in the SITE-BASIS
!
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      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 ccccccccccccccccccccc
        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 ccccccccccccccccccccc
         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 ccccccccccccccccccccc
            do k = 1, chain_size-1
              IF(bip(k)*bip(k+1).EQ.1) then
                VecI(i,j)=VecI(i,j)+Jxy/2.
                VecI(j,i)=VecI(j,i)+Jxy/2.
              ENDIF
            enddo

!ccccccccccccccc NNN ccccccccccccccccccccc
            do k = 1, chain_size-2
              IF(bip(k)*bip(k+2).EQ.1) then
                VecI(i,j)=VecI(i,j)+alphaI*Jxy/2.
                VecI(j,i)=VecI(j,i)+alphaI*Jxy/2.
              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
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!
! SUBROUTINE to write the HAMILTONIAN in the SITE-BASIS
!
! and also to get the elements of the observables
!
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine HamiltonianFINAL()

      use variables
      implicit none

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


!c INITIALIZATION
      Do i=1,dimTotal
        Do j=1,dimTotal
          VecF(i,j)=0.0d0
        Enddo
        IEdiag(i)=0.0d0
        Czzaux(i)=0.0d0
        Czza2(i)=0.0d0
        Czza3(i)=0.0d0
      Enddo

      Do j=1,chain_size-1
        Npair(j)=0
      Enddo

      nxx=0
      nxx2=0
      nxx3=0
      mid=chain_size/2

! 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))

! To compute Czz for sites (L/2,L/2+1), (L/2,L/2+2), (L/2,L/2+)3
        Czzaux(i)=Czzaux(i)+(0.25d0)*(-1.0d0)**(site_basis(i,mid)+site_basis(i,mid+1))
        Czza2(i)=Czza2(i)+(0.25d0)*(-1.0d0)**(site_basis(i,mid)+site_basis(i,mid+2))
        Czza3(i)=Czza3(i)+(0.25d0)*(-1.0d0)**(site_basis(i,mid)+site_basis(i,mid+3))

!ccccccccccccccc NN ccccccccccccccccccccc
        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))
          IEdiag(i)=IEdiag(i)+(JzF/4.d0)*(-1.0d0)**(site_basis(i,j)+site_basis(i,j+1))
        enddo

!ccccccccccccccc NNN ccccccccccccccccccccc
         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))
       IEdiag(i)=IEdiag(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
            bip(k) = mod(site_basis(i,k) + site_basis(j,k),2)
            tot = tot + bip(k)
          enddo

          IF(tot.EQ.2) then

! To compute Cxx for sites L/2,L/2+1
                IF(bip(mid)*bip(mid+1).EQ.1) then
                  nxx=nxx+1
                  cxxI(nxx)=i
                  cxxJ(nxx)=j
                ENDIF
! To compute Cxx for sites L/2,L/2+2
                IF(bip(mid)*bip(mid+2).EQ.1) then
                  nxx2=nxx2+1
                  cxxI2(nxx2)=i
                  cxxJ2(nxx2)=j
                ENDIF
! To compute Cxx for sites L/2,L/2+3
                IF(bip(mid)*bip(mid+3).EQ.1) then
                  nxx3=nxx3+1
                  cxxI3(nxx3)=i
                  cxxJ3(nxx3)=j
                ENDIF

!ccccccccccccccc NN ccccccccccccccccccccc
```

```
            do k = 1, chain_size-1
              IF(bip(k)*bip(k+1).EQ.1) then
                VecF(i,j)=VecF(i,j)+Jxy/2.
                VecF(j,i)=VecF(j,i)+Jxy/2.
                Npair(1)=Npair(1)+1
                pairI(1,Npair(1))=i
                pairJ(1,Npair(1))=j
              ENDIF
            enddo

!ccccccccccccccc NNN ccccccccccccccccccccc
            do k = 1, chain_size-2
              IF(bip(k)*bip(k+2).EQ.1) then
                VecF(i,j)=VecF(i,j)+alphaF*Jxy/2.
                VecF(j,i)=VecF(j,i)+alphaF*Jxy/2.
                Npair(2)=Npair(2)+1
                pairI(2,Npair(2))=i
                pairJ(2,Npair(2))=j
              ENDIF
            enddo
!ccccccccccccccc DISTANT PAIRS ccccccccccc
         DO jk=3,chain_size-1
           do k=1,chain_size-jk
             IF(bip(k)*bip(k+jk).EQ.1) then
               Npair(jk)=Npair(jk)+1
               pairI(jk,Npair(jk))=i
               pairJ(jk,Npair(jk))=j
             ENDIF
           enddo
         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 HAMILTONIAN
!  in the SITE-BASIS
      return
      end subroutine HamiltonianFINAL
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!
! SUBROUTINE to compute IPR and Shannon entropy in the XXZ basis
!
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine IPRShannon()

      use variables
      implicit none
```

```
!c Multiplying matrices
       allocate(Haux(dimTotal,dimTotal))
       allocate(HamMF(dimTotal,dimTotal))
       call DGEMM('n','n',dd,dd,dd,1.0d0,VecF,dd,VecI,dd,0.0d0,Haux,dd)
       call DGEMM('t','n',dd,dd,dd,1.0d0,VecI,dd,Haux,dd,0.0d0,HamMF,dd)
       deallocate(Haux)

       CALL DSYEV('V','U',dimTotal,HamMF,dimTotal,EigF,WORK,7*dimTotal,INFO)

       write(17,300) '#','Energy','IPR','Shannon'
       Do k=1, dimTotal
         auxIPR=0.0d0
         Shan=0.0d0
         Do j=1,dimTotal
           auxIPR=auxIPR+HamMF(j,k)**4
           IF(HamMF(j,k)**2.gt.0.0d0) then  ! to avoid problems with log(negative)
           Shan=Shan-(HamMF(j,k)**2)*log(HamMF(j,k)**2)
           ENDIF
         Enddo
         IPR=1.0d0/auxIPR
         write(17,301) EigF(k), IPR, Shan
       Enddo
       deallocate(HamMF)

301    format(3E18.9)
300    format(A1,1X,A13,3X,A13,5X,A13)

!  END of SUBROUTINE that computes IPR and Shannon
!  in the XXZ-BASIS
      return
      end subroutine IPRShannon
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!
! SUBROUTINE to select an INITIAL STATE according to TEMPERATURE
!
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine InitialStateXXZ()

      use variables
      implicit none

! Value of the energy for the chosen temperature
       aux=0.0d0
       partition=0.0d0
       Do k=1,dimTotal
          aux=aux+EigF(k)*exp(-EigF(k)/Temperature)
          partition=partition+exp(-EigF(k)/Temperature)
       Enddo
       Etemp=aux/partition
! SELECTING the initial state according to the FINAL Hamiltonian
!ccccccccccccc DIAGONAL ENSEMBLE ccccccccccccccccccccccccccccccccccccccccccccccc
```

```fortran
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Project VecF on the eigenstates of the FINAL Hamiltonian
! Look for <ini| Hfinal |ini> closest to Etemp above
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
       menor=1000000.0
       Do j=1,dimTotal
          Do k=1,dimTotal
            auxIni(k)=VecI(k,j)
          Enddo
       call DGEMV('t',dd,dd,1.0d0,VecF,dd,auxIni,1,0.0d0,CalphaAUX,1)
          auxEn=0
          Do k=1,dimTotal
            auxEn=auxEn+(CalphaAUX(k)**2)*EigF(k)
          Enddo
          IF (dabs(Etemp-auxEn ).LT.menor ) then
             menor=dabs(Etemp-auxEn )
             qual=j
             Einitial=auxEn
             Calpha=CalphaAUX
          ENDIF
       Enddo
! The INITIAL state and its actual energy is then
       Do k=1,dimTotal
          Initial(k)=VecI(k,qual)
       Enddo
       EfromHI=EigI(qual)
! Squared width of the energy shell
       width=0.0d0
       Do k=1,dimTotal
         width=width+(Calpha(k)**2)*(EigF(k)-Einitial)**2
       Enddo

! cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! Output with information about the initial state
       write(40,*) 'Temperature', Temperature
       write(40,*)
       write(40,*) 'Canonical energy for chosen Temperature', Etemp
       write(40,*)
       write(40,*) 'Energy of the initial state with respect to the final Ham'
       write(40,*) '<ini| Hfinal |ini> = ', Einitial
       write(40,*)
       write(40,*) 'Energy of the initial state with respect to the initial Ham', EfromHI
       write(40,*) '<ini| Hinitial |ini> = ', EfromHI
       write(40,*)
       write(40,*) 'Variance of the Energy Shell', width
! cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

!  END of SUBROUTINE that selects the initial state
!  according to Temperature
      return
      end subroutine InitialStateXXZ
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!
!           SUBROUTINE to compute the OBSERVABLES
!
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine Observables()

      use variables
      implicit none

! 2 and 3 below refers to 2nd and 3rd neighbors
! INITIALIZATION ccccccccccccccccc
! microcanonical average
       numMIC=0

       IEmic=0.0d0
       KEmic=0.0d0
       Czzmic=0.0d0
       Czz2mic=0.0d0
       Czz3mic=0.0d0
       Cxxmic=0.0d0
       Cxx2mic=0.0d0
       Cxx3mic=0.0d0
! diagonal ensemble average
       IEde=0.0d0
       KEde=0.0d0
       Czzde=0.0d0
       Czz2de=0.0d0
       Czz3de=0.0d0
       Cxxde=0.0d0
       Cxx2de=0.0d0
       Cxx3de=0.0d0
! Magnetization and Sts
       Do j=1,chain_size
         SiteMagMIC(j)=0.0d0
         SiteMagDE(j)=0.0d0
       Enddo
       Do k=0,chain_size/2
         StzzMIC(k)=0.0d0
         StxxMIC(k)=0.25d0
         StzzDE(k)=0.0d0
         StxxDE(k)=0.25d0
       Enddo
! ccccccccccccccccccccccccccccccc



! ccccccccccccccccccc LOOP for all EIGENSTATES ccccccccccccccccccccccccc
! ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       write(30,200) '#','Energy','KE','IE'
       write(32,210) '#','Energy','Cz(L/2,L/2+1)', 'Cz(L/2,L/2+2)','Cz(L/2,L/2+3)'
       write(34,210) '#','Energy','Cx(L/2,L/2+1)', 'Cx(L/2,L/2+2)','Cx(L/2,L/2+3)'
       write(36,220) '#','Energy','Sz(0), Sz(1), ... Sz(L/2)'
       write(38,220) '#','Energy','Sx(0), Sx(1), ... Sx(L/2)'
```

```
       Do kk=1,dimTotal

! INITIALIZATION for the OBSERVABLES for each EIGENSTATE ccccccccccccccc
       IE=0.0d0
       KE=0.0d0
       Czz=0.0d0
       Czz2=0.0d0
       Czz3=0.0d0
       Cxx=0.0d0
       Cxx2=0.0d0
       Cxx3=0.0d0
       Do j=1,chain_size
         SiteMag(j)=0.0d0
       Enddo
       Do k=0,chain_size/2
         Stzz(k)=0.0d0
         Stxx(k)=0.25d0
       Enddo
! ccccccccccccccccccccccccccccccc


! OBSERVABLES for each eigenstate kk ccccccccccccccccccccccccccccccccccc

       DO i=1, dimTotal

! IE, site magnetization, and Stzz are DIAGONAL, so we just need aux, that is
!                                                the weight of each basis vector i
         aux=VecF(i,kk)**2

! IE
         IE=IE+aux*IEdiag(i)
! Czz (L/2,L/2+1), (L/2,L/2+2), (L/2,L/2+3)
         Czz=Czz+aux*Czzaux(i)
         Czz2=Czz2+aux*Czza2(i)
         Czz3=Czz3+aux*Czza3(i)
! SITE MAGNETIZATION
         Do j=1,chain_size
           phaseMag=(-1.0d0)**(1+site_basis(i,j))
           SiteMag(j)=SiteMag(j)+aux*phaseMag
         Enddo
! Stzz
         Do k=0,chain_size/2
           ZZ(k)=0.25d0
           Do j=1,chain_size-1
             phase=(0.5d0/dble(chain_size))*dcos(pi*dble(2*k*j)/dble(chain_size))
                       auxZ=0.0d0
                        do jj=1,chain_size-j
                          auxZ=auxZ+(-1.0d0)**(site_basis(i,jj)+site_basis(i,jj+j))
                        enddo
                        ZZ(k)=ZZ(k)+phase*auxZ
           Enddo
           Stzz(k)=Stzz(k)+aux*ZZ(k)
         Enddo
       ENDDO


! The next observables are NOT DIAGONAL, they involve different basis vectors
! NOTE: 0.5 =(1/4)*2
```

```
!       it appears because Sx.Sx = 1/4(sigma_x.sigma_x) and we need pairs i,j and also j,i
!Cxx  (L/2,L/2+1)
       Do j=1,nxx
        Cxx=Cxx+0.5*VecF(cxxI(j),kk)*VecF(cxxJ(j),kk)
       Enddo
!Cxx  (L/2,L/2+2)
       Do j=1,nxx2
        Cxx2=Cxx2+0.5*VecF(cxxI2(j),kk)*VecF(cxxJ2(j),kk)
       Enddo
!Cxx  (L/2,L/2+3)
       Do j=1,nxx3
        Cxx3=Cxx3+0.5*VecF(cxxI3(j),kk)*VecF(cxxJ3(j),kk)
       Enddo
! KE
! NOTE: it is 2(VecF(I)*VecF(J))(Jxy/2), so the 2's cancel
! the reason why it is 2(...) is that we have i with j, but also j with i
! that is 1 with 2, but also 2 with 1, which is not taken into account
! in the subroutine HamiltonianFINAL
       Do j=1,Npair(1)
         KE=KE+VecF(pairI(1,j),kk)*VecF(pairJ(1,j),kk)
       Enddo
       Do j=1,Npair(2)
         KE=KE+alphaF*VecF(pairI(2,j),kk)*VecF(pairJ(2,j),kk)
       Enddo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Stxx
       Do k=0,chain_size/2
         Do j=1,chain_size-1
           phase=(0.5d0/dble(chain_size))*dcos(pi*dble(2*k*j)/dble(chain_size))
                 auxX=0.0d0
                 Do jj=1,Npair(j)
         auxX=auxX+2.0d0*VecF(pairI(j,jj),kk)*VecF(pairJ(j,jj),kk)
           Enddo
           Stxx(k)=Stxx(k)+phase*auxX
         Enddo
       Enddo

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


!!!!!!!!!!! WRITE THE OBSERVABLES in FILES !!!!!!!!!!!!!!!!
       write(30,117) EigF(kk), KE,IE
       write(32,118) EigF(kk), Czz,Czz2,Czz3
       write(34,118) EigF(kk), Cxx,Cxx2,Cxx3
       write(36,119) EigF(kk), (Stzz(j), j=0,chain_size/2)
       write(38,119) EigF(kk), (Stxx(j), j=0,chain_size/2)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!



!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!          DIAGONAL and MICROCANONICAL ensembles
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Diagonal
       IEde = IEde+IE*dabs(Calpha(kk))**2
       KEde = KEde+KE*dabs(Calpha(kk))**2
```

```
        Czzde=Czzde+Czz*dabs(Calpha(kk))**2
        Czz2de=Czz2de+Czz2*dabs(Calpha(kk))**2
        Czz3de=Czz3de+Czz3*dabs(Calpha(kk))**2

        Cxxde=Cxxde+Cxx*dabs(Calpha(kk))**2
        Cxx2de=Cxx2de+Cxx2*dabs(Calpha(kk))**2
        Cxx3de=Cxx3de+Cxx3*dabs(Calpha(kk))**2

        Do k=0,chain_size/2
          Stzzde(k)=Stzzde(k)+Stzz(k)*dabs(Calpha(kk))**2
          Stxxde(k)=Stxxde(k)+Stxx(k)*dabs(Calpha(kk))**2
        Enddo

! Microcanonical
        delta=dabs(EigF(kk)-Einitial)
          if(delta.le.deltaMIC/2.0d0) then
            IEmic = IEmic+IE
            KEmic = KEmic+KE

            Czzmic=Czzmic+Czz
            Czz2mic=Czz2mic+Czz2
            Czz3mic=Czz3mic+Czz3

            Cxxmic=Cxxmic+Cxx
            Cxx2mic=Cxx2mic+Cxx2
            Cxx3mic=Cxx3mic+Cxx3

            Do k=0,chain_size/2
          StzzMIC(k)=StzzMIC(k)+Stzz(k)
          StxxMIC(k)=StxxMIC(k)+Stxx(k)
            Enddo

            numMIC=numMIC+1
          endif

! ccccccccccccccccccc CLOSE LOOP for all EIGENSTATES ccccccccccccccccccc
! ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        ENDDO
        mMI=float(numMIC)
! OUTPUT files
        write(20,214) '#','Temperature', 'IEde', 'IEmic', 'KEde', 'KEmic'
        write(20,120) Temperature,IEde,IEmic/mMI,KEde,KEmic/mMI

        write(22,212) '#','Temperature', 'Czde', 'Cz2de', 'Cz3de'
        write(22,122) Temperature,Czzde,Czz2de,Czz3de
        write(22,212) '#','Temperature', 'Czmic', 'Cz2mic', 'Cz3mic'
        write(22,122) Temperature,Czzmic/mMI,Czz2mic/mMI,Czz3mic/mMI

        write(24,212) '#','Temperature', 'Cxde', 'Cx2de', 'Cx3de'
        write(24,122) Temperature,Cxxde,Cxx2de,Cxx3de
        write(24,212) '#','Temperature', 'Cxmic', 'Cx2mic', 'Cx3mic'
        write(24,122) Temperature,Cxxmic/mMI,Cxx2mic/mMI,Cxx3mic/mMI

        write(26,216) '#','Temperature', 'StzDE(0), StzDE(1), ... StzDE(L/2)'
        write(26,124) Temperature,(Stzzde(k), k=0,chain_size/2)
        write(26,216) '#','Temperature', 'StzME(0), StzME(1), ... StzME(L/2)'
        write(26,124) Temperature,(StzzMIC(k), k=0,chain_size/2)
```

```
       write(28,216) '#','Temperature', 'StxDE(0), StxDE(1), ... StxDE(L/2)'
       write(28,124) Temperature,(Stxxde(k), k=0,chain_size/2)
       write(28,216) '#','Temperature', 'StxME(0), StxME(1), ... StxME(L/2)'
       write(28,124) Temperature,(StxxMIC(k), k=0,chain_size/2)


! FORMAT
117    format(3E18.9)
118    format(4E18.9)
119    format(10E18.9)

120    format(F14.4,4E18.9)
122    format(F14.4,3E18.9)
124    format(F14.4,10E18.9)

200    format(A1,1X,A13,3X,A13,5X,A13)
210    format(A1,1X,A13,7X,A13,5X,A13,5X,A13)
220    format(A1,1X,A13,7X,A25)
212    format(A1,1X,A13,1X,A13,5X,A13,5X,A13)
214    format(A1,1X,A13,1X,A13,5X,A13,5X,A13,5X,A13)
216    format(A1,1X,A13,1X,A35)


!  END of SUBROUTINE that computes OBSERVABLES
      return
      end subroutine Observables
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```