```
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! This code gives us the eigenstates of the XXZ model for an open
! spin-1/2 chain of size L and a certain fixed number of up-spins
! in the SITE-BASIS.
!
! It also gives the eigenstates of the XX model in the SITE-BASIS.
! These we use to transform the eigenstates of the XXZ model into the
! basis of the XX model.
!
! IPR for the eigenstates of the XXZ model in the SITE-basis and in the
! XX-basis are computed.
!
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

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

       module variables

       implicit none

       integer (kind=4) :: i,j,k,kj,ii
       integer (kind=4) :: chain,upspins
       integer (kind=4) :: dimTotal,dd
       integer (kind=4) :: facL,facU,facLU


! PARAMETERS of the XX Hamiltonian
       real (kind=8) :: JxyI,JzI
! PARAMETERS of the XXZ Hamiltonian
       real (kind=8) :: JxyF,JzF
       integer (kind=4) :: Jztot


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


! Eigenvectors and eigenvalues of the XX Hamiltonian
       real (kind=8), dimension(:), allocatable :: EigI
       real (kind=8), dimension(:,:), allocatable :: VecI
! Eigenvectors and eigenvalues of the XXZ Hamiltonian (SITE-BASIS)
       real (kind=8), dimension(:), allocatable :: EigF
       real (kind=8), dimension(:,:), allocatable :: VecF
! Eigenvectors of the XXZ Hamiltonian                 (XX-BASIS)
```

```
      real (kind=8), dimension(:,:), allocatable :: VecMF


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

! IPR
      real (kind=8) :: IPRsite,aveIPRsite
      real (kind=8) :: IPRxx,aveIPRxx
      real (kind=8) :: aux

! For the OUTPUT files
      character(len=70) saida


      end module


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!             Program starts here
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      Program IPRbasis

      use variables
      implicit none


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! PARAMETERS
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      chain=10
      upspins=chain/2
      call SectorDimension()
      dimTotal=facL/facU
      dd=dimTotal

! XX model
      JxyI=1.0d0
      JzI=0.0d0
! XXZ model
      JxyF=1.0d0
      Jztot=51
```

```
! Output File   cccccccccccccccccccc
       saida='IPRsiteIPRxx_XXZmodel.dat'
       OPEN(unit=20, FILE=saida,STATUS='UNKNOWN')


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c CREATE SITE-BASIS
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       allocate(basis(dimTotal,chain))
       call SiteBasis()


       allocate(work(7*dimTotal))
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! EIGENVALUES and EIGENSTATES of the XX model
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       allocate(EigI(dimTotal))
       allocate(VecI(dimTotal,dimTotal))
       call HamiltonianXX()
       CALL DSYEV('V','U',dimTotal,VecI,dimTotal,EigI,WORK,7*dimTotal,INFO)



!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! LOOP for the values of JzF
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       allocate(EigF(dimTotal))
       allocate(VecF(dimTotal,dimTotal))
       allocate(VecMF(dimTotal,dimTotal))
       write(20,110) 'DELTA','IPR in site','IPR in XX'

       Do kj=1,Jztot
         JzF=0.5d0*dfloat(kj-1)
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! EIGENVALUES and EIGENSTATES of the XXZ model
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       call HamiltonianXXZ()
       CALL DSYEV('V','U',dimTotal,VecF,dimTotal,EigF,WORK,7*dimTotal,INFO)


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
!c IPR for the XXZ eigenstates in the SITE-BASIS
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       aveIPRsite=0.0d0
       DO i=1,dimTotal
         aux=0.0d0
         Do j=1,dimTotal
           aux=aux+VecF(j,i)**4
         Enddo
         IPRsite=1.0d0/aux
         aveIPRsite=aveIPRsite+IPRsite
       ENDDO
       aveIPRsite=aveIPRsite/dfloat(dimTotal)


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c IPR for the XXZ eigenstates in the XX-BASIS
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       call DGEMM('t','n',dd,dd,dd,1.0d0,VecI,dd,VecF,dd,0.0d0,VecMF,dd)

       aveIPRxx=0.0d0
       DO i=1,dimTotal
         aux=0.0d0
         Do j=1,dimTotal
           aux=aux+VecMF(j,i)**4
         Enddo
         IPRxx=1.0d0/aux
         aveIPRxx=aveIPRxx+IPRxx
       ENDDO
       aveIPRxx=aveIPRxx/dfloat(dimTotal)

!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! OUTPUT
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        write(20,120) JzF,aveIPRsite,aveIPRxx


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! CLOSING the LOOP for Jz
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       ENDDO
110    Format (3X,A5,5X,A11,6X,A9)
120    Format (F8.3,2E18.9)
```

```
      close(120)


!cccccccccccccccccccccccccccccccccccccccccccccccc
!c END END END END END END END END END END END END
!cccccccccccccccccccccccccccccccccccccccccccccccc
!      STOP
      END


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccc SUBROUTINES SUBROUTINES SUBROUTINES cccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc




!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! *************** DIMENSION OF SUBSPACE ***************************
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SectorDimension()
      use variables
      implicit none

      facL=1
      Do i=chain-upspins+1,chain
        facL=facL*i
      Enddo

      facU=1
      Do i=2,upspins
        facU=facU*i
      Enddo



!c END of SUBROUTINE for DIMENSION OF SUBSPACE
      return
      end subroutine SectorDimension
```

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

      INTEGER (kind=4) :: ib,jb

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

      mtc=.false.

!c INITIALIZATION
      Do ib=1,dimTotal
        Do jb=1,chain
          basis(ib,jb)=0
        enddo
      enddo

      ii=1

71   call NEXKSB(chain,upspins,in,mtc,m2,h)
      Do jb=1,upspins
        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 XX HAMILTONIAN in the SITE-BASIS
!
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       subroutine HamiltonianXX()

       use variables
       implicit none

       INTEGER (kind=4) :: tot,DifferentSite(chain)

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

! DIAGONAL ELEMENTS
************************************************************
       Do i=1,dimTotal
!cccccccccccccccc NN cccccccccccccccccccccc
         Do j=1,chain-1
       VecI(i,i)=VecI(i,i)+(JzI/4.d0)*(-1.0d0)**(basis(i,j)+basis(i,j+1))
         enddo
!cccccccccccccccccccccccccccccccccccccccccc
```

```
! 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
            DifferentSite(k)=0
          Enddo

          Do k = 1, chain
            If( basis(i,k).ne.basis(j,k) ) then
            tot = tot + 1
            DifferentSite(tot)=k
            Endif
          Enddo

          IF(tot.EQ.2) then
!cccccccccccccc NN cccccccccccccccccccccccccccccc
          IF((DifferentSite(2) - DifferentSite(1)).EQ.1) then
            VecI(i,j)=VecI(i,j)+JxyI/2.0d0
            VecI(j,i)=VecI(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 XX HAMILTONIAN
!  in the SITE-BASIS for a FIXED NUMBER of UP-SPINS
       return
       end subroutine HamiltonianXX
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!
! SUBROUTINE to write the XXZ HAMILTONIAN in the SITE-BASIS
!
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       subroutine HamiltonianXXZ()

       use variables
       implicit none

       INTEGER (kind=4) :: tot,DifferentSite(chain)

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

! DIAGONAL ELEMENTS
*************************************************************
       Do i=1,dimTotal
!cccccccccccccc NN cccccccccccccccccccc
         Do j=1,chain-1
       VecF(i,i)=VecF(i,i)+(JzF/4.d0)*(-1.0d0)**(basis(i,j)+basis(i,j+1))
         enddo
!cccccccccccccccccccccccccccccccccccccccc
! 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
            DifferentSite(k)=0
          Enddo
```

```
          Do k = 1, chain
            If( basis(i,k).ne.basis(j,k) ) then
            tot = tot + 1
            DifferentSite(tot)=k
            Endif
          Enddo

          IF(tot.EQ.2) then
!cccccccccccccc NN cccccccccccccccccccccccccccccc
          IF((DifferentSite(2) - DifferentSite(1)).EQ.1) then
            VecF(i,j)=VecF(i,j)+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 XXZ HAMILTONIAN
!  in the SITE-BASIS for a FIXED NUMBER of UP-SPINS
       return
       end subroutine HamiltonianXXZ
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```