```
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! This code gives us the eigenvalues and eigenstates of the XXZ model for a
! spin-1/2 chain of size L and a certain fixed number of up-spins.
!
! In the code: L is called "chain", the dimension is "dimTotal", and the
! site-basis vectors are "basis".
!
! The code also gives the eigenvalues of the parity operator and the
operator
! that performs a pi-rotation around the x axis.
!
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

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

       module variables

       implicit none

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

! PARAMETERS of the XXZ Hamiltonian
       real (kind=8) :: JxyI,JzI

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

! Eigenvectors and eigenvalues of the XXZ Hamiltonian
       real (kind=8), dimension(:), allocatable :: EigI
       real (kind=8), dimension(:,:), allocatable :: VecI

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

! SYMMETRIES
       integer (kind=4), dimension(:), allocatable :: parity,rotationx


       end module
```

```
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!             Program starts here
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       Program XXZModels

       use variables
       implicit none


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! PARAMETERS
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       chain=6
       upspins=chain/2
       call SectorDimension()
       dimTotal=facL/facU

       JxyI=1.0d0
       JzI=0.4d0


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


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


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c SYMMETRIES
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       allocate(parity(dimTotal))
       call ParityValues()
```

```
       allocate(rotationx(dimTotal))
       call RotationXValues()


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! OUTPUT
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       write(*,*)
       write(*,20) 'EIGENVALUES','PARITY','ROTATIONx'
20    Format (4X,A11,5X,A6,4X,A9)
       Do i=1,dimTotal
         write(*,30) EigI(i),parity(i),rotationx(i)
       Enddo
30    Format (E18.9,I6,I10)



!cccccccccccccccccccccccccccccccccccccccccccccccccc
!c END END END END END END END END END END END END
!cccccccccccccccccccccccccccccccccccccccccccccccccc
!      STOP
       END



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccc SUBROUTINES SUBROUTINES SUBROUTINES ccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc




!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! *************** DIMENSION OF SUBSPACE ***************************
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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





!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! *************** WRITING THE SITE-BASIS **************************
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
!cccccccccccccccccccccccccccccccccccccccccc
!c SUBROUTINE to get the COMBINATIONS
!cccccccccccccccccccccccccccccccccccccccccc
      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
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!
! SUBROUTINE for the PARITY PAIRS
!
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ParityValues()

      use variables
      implicit none

      real (kind=8) :: proje
      INTEGER (kind=4) :: ref
      INTEGER (kind=4), dimension(:), allocatable :: pair

      allocate(pair(dimTotal))
```

```
! Identify the PAIRS
      Do i=1,dimTotal
        Do j=1,dimTotal
                        ref=0
                Do k=1,chain
                        ref = ref + mod(basis(i,k) + basis(j,chain+1-k),2)
                enddo
                if(ref.eq.0) then
                pair(i)=j
                endif
        enddo
      enddo


! Find PARITY
      Do i=1,dimTotal
        proje=0.0d0
        Do j=1,dimTotal
          proje=proje+0.25d0*(VecI(j,i)+VecI(pair(j),i))**2
        Enddo
        If(1.0d0 - proje.lt.0.001) then
          parity(i)=1
        Else
          parity(i)=-1
        Endif
      Enddo

        do i=1,dimTotal
        enddo

      deallocate(pair)
!c END of SUBROUTINE for PARITY values
      return
      end subroutine ParityValues
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!
! SUBROUTINE for the Pi-rotation in x SYMMETRY
!
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine RotationXValues()
```

```
      use variables
      implicit none

      real (kind=8) :: rotrot
      INTEGER (kind=4) :: rotx
      INTEGER (kind=4), dimension(:), allocatable :: pair
      allocate(pair(dimTotal))


! Identify the pairs under the pi-rotation in x
      DO i=1,dimTotal
        Do j=1,dimTotal
          rotx=0
          do k=1,chain
      rotx = rotx + mod(basis(i,k) + basis(j,k),2)
          enddo
          if(rotx.eq.chain) then
                pair(i)=j
          endif
        Enddo
      ENDDO

! Find the eigenvalues of the Pi-rotation in x operator
      Do i=1,dimTotal
        rotrot=0.0d0
        Do j=1,dimTotal
          rotrot=rotrot+0.25d0*(VecI(j,i)+VecI(pair(j),i))**2
        Enddo
        If(1.0d0-rotrot.lt.0.001) then
          rotationx(i)=1
        Else
          rotationx(i)=-1
        Endif
      Enddo


      deallocate(pair)
!c END of SUBROUTINE for RotationX values
      return
      end subroutine RotationXValues


!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
           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 XXZ HAMILTONIAN
!  in the SITE-BASIS for a FIXED NUMBER of UP-SPINS
       return
       end subroutine HamiltonianXXZ
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```