```
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! This code gives us the eigenvalues of spin-1/2 models with OPEN
! boundary conditions and different parameters.
!
! The eigenvalues are separated according to parity.
! The chain has an even number of sites. Parity is an issue only when
! there is no impurity at the edge or in the middle of the chain.
!
! This code does not give the level spacing distribution. But with the
! eigenvalues separated by parity, this can be done.
!
!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 spin-1/2 Hamiltonian
! NN couplings
       real (kind=8), PARAMETER :: JxyI=1.0d0
       real (kind=8), PARAMETER :: JzI=0.5d0
! NNN couplings
       real (kind=8), PARAMETER :: lambdaI=1.0d0
       real (kind=8), PARAMETER :: JxyInnn=1.0d0
       real (kind=8), PARAMETER :: JzInnn=0.5d0
! impurities
       real (kind=8), PARAMETER :: edgeI=0.0d0
       real (kind=8), PARAMETER :: defI=0.0d0

! 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

! For the OUTPUT file
       character(len=70) sai
       character(len=2) LC
```

```
       character(len=4) DeI,DeJ,XYI,alp,imp,bd


       end module


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!             Program starts here
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       Program EigenvaluesNNN

       use variables
       implicit none


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! PARAMETERS
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       chain=18
       upspins=chain/3
       call SectorDimension()
       dimTotal=facL/facU


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! OUTPUT FILES
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      write(LC,9) chain
9    format(i2.2)
      write(DeI,10) JzI
!
      write(alp,10) lambdaI
      write(DeJ,10) JzInnn
      write(XYI,10) JxyInnn
!
      write(imp,10) defI
      write(bd,10) edgeI
10   format(F4.2)

      sai='EigL'//LC//'Jz'//DeI//'la'//alp//'JJx'//XYI//'JJz'//DeJ//'b'//bd//'d'//imp//'.dat'
       OPEN(unit=30, FILE=sai,STATUS='UNKNOWN')



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


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! EIGENVALUES and EIGENSTATES of the spin-1/2 model
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

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


!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c SYMMETRIES: parity is an issue ONLY in the absence of defects
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      if(defI.eq.0.0d0) then
      if(edgeI.eq.0.0d0) then
      allocate(parity(dimTotal))
      call ParityValues()
      endif
      endif

!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! OUTPUT
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      if(defI.eq.0.0d0) then
      if(edgeI.eq.0.0d0) then
      write(30,120) '#','VECTOR','PARITY','EIGENVALUES'
120   Format (A1,1X,A6,1X,A6,4X,A11)
      Do i=1,dimTotal
        write(30,130) i, parity(i), EigI(i)
      Enddo
130   Format (2I6,E20.9)
      endif
      endif

      if(defI.gt.0.0d0 .OR. edgeI.gt.0.0d0) then
      write(30,121) '#','VECTOR','EIGENVALUES'
121   Format (A1,1X,A6,4X,A11)
      Do i=1,dimTotal
        write(30,131) i, EigI(i)
      Enddo
131   Format (I6,E20.9)
      endif


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



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccc SUBROUTINES SUBROUTINES SUBROUTINES ccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! *************** DIMENSION OF SUBSPACE ***************************
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
!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
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc




!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!
! SUBROUTINE for the PARITY PAIRS
!
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
!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)


!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 at the EDGE on SITE 1 cccccccccccccc
         VecI(i,i)=VecI(i,i)+0.5d0*edgeI*(-1.0d0)**(1+basis(i,1))

!cccccccc DEFECT at the MIDDLE ccccccccccccccc
         VecI(i,i)=VecI(i,i)+0.5d0*defI*(-1.0d0)**(1+basis(i,chain/2))

!ccccccccccccccc 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

!ccccccccccccccc NNN cccccccccccccccccccccc
         Do j=1,chain-2
       VecI(i,i)=VecI(i,i)+lambdaI*(JzInnn/4.d0)*(-1.0d0)**(basis(i,j)+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
             bip(k) = mod(basis(i,k) + basis(j,k),2)
             tot = tot + bip(k)
           enddo

           IF(tot.EQ.2) then

!ccccccccccccccc NN cccccccccccccccccccccc
           do k = 1, chain-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

!ccccccccccccccc NNN cccccccccccccccccccccc
           do k = 1, chain-2
             IF(bip(k)*bip(k+2).EQ.1) then
               VecI(i,j)=VecI(i,j)+lambdaI*JxyInnn/2.0d0
               VecI(j,i)=VecI(j,i)+lambdaI*JxyInnn/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
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```