```
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! This code does the following:
!
!   1) Writes the NNN and the impurity model Hamiltonian matrices in the basis
! corresponding to eigenstates of the XXZ model.
!
!   2) It orders the diagonal elements of these matrices from smallest to
! largest value using MRGRNK.
!
!   3) The columns of the output file gives
!(i)   diagonal elements of the matrices, [For Fig.9(a)]
!(ii)  number of non-zero off-diagonal elements in each row
!      (CONNECTIVITY) [For Fig.9(b)]
!(iii) variance of the absolute value of all off-diagonal elements
!      which is used as a threshold for selecting the non-zero
!      off-diagonal elements [For item (ii)]
!(iv)  average strength of the absolute value of the off-diagonal elements
!      [For Fig.9(d), it is the v_n]
!(v)   mean level spacing between directly coupled states [For Fig.9(d)]
!(vi)  average of the absolute values of the off-diagonal elements as they
!      get away from the diagonal [For Fig.9(c)]
!
!
!
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

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

       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

! PARAMETERS of the Hamiltonian
       real (kind=8) :: Jxy,Jz,defborder,alphaI,alphaF
       real (kind=8) :: defectI,defectF

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

! HAMILTONIAN=EIGENSTATES of the MF-BASIS (unperturbed vectors)
! and UNPERTURBED eigenvalues
       real (kind=8), dimension(:,:), allocatable :: UnpVecSite
       real (kind=8), dimension(:), allocatable :: UnpEig

! HAMILTONIAN=EIGENSTATES in the SITE-BASIS
       real (kind=8), dimension(:,:), allocatable :: PertVecSite

! EIGENSTATES in the MF-BASIS
       real (kind=8), dimension(:,:), allocatable :: HamMF,Haux
       real (kind=8) :: aux,offi
```

```
       real (kind=8), dimension(:), allocatable :: XVALT
       integer (kind=4), dimension(:), allocatable :: IMULT

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

! For the OUTPUT files
       character(len=1) up
       character(len=2) LC
       character(len=4) JzC,alC,deC,Eni
       character(len=55) saiEF,saiLDOS,saiMF,saiCO

       end module


!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!             Program starts here
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       Program Model1Model2

       use variables
       implicit none

! Dimension of the system
       integer (kind=4) :: BINOMIAL

! SIGMA for the SHELL
       INTEGER (kind=4) :: connectMF,tot
       real (kind=8) :: square,var,ave,mV,maxi,mini
       real (kind=8) :: divi

! for CHANGING the basis of the Hamiltonian
       integer (kind=4) :: dd
       real (kind=8), dimension(:,:), allocatable :: r1

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! PARAMETERS
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

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

       chain_size=18
       upspins=6
       dimTotal=18564

!       chain_size=6
!       upspins=2
!       dimTotal=15

       Jxy=1.0d0
       defborder=0.1d0
       defectI=0.0d0
       alphaI=0.0d0
```

```
      write(LC,8) chain_size
8     format(i2.2)
      write(up,9) upspins
9     format(i1)


!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c CREATE SITE-BASIS
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       allocate(site_basis(dimTotal,chain_size))
       call SiteBasis()


       Do kk=1,2
         if(kk.eq.1) then
           Jz=0.48d0
          defectF=0.00d0
          alphaF=0.48d0
         endif
         if(kk.eq.2) then
           Jz=0.48d0
          defectF=0.54d0
          alphaF=0.0d0
         endif
         if(kk.eq.3) then
           Jz=0.48d0
          defectF=0.00d0
          alphaF=0.76d0
         endif
         if(kk.eq.4) then
           Jz=0.94d0
          defectF=0.76d0
          alphaF=0.0d0
         endif
         if(kk.eq.5) then
           Jz=0.94d0
          defectF=1.08d0
          alphaF=0.0d0
         endif
         if(kk.eq.6) then
           Jz=0.9d0
          defectF=1.08d0
          alphaF=0.0d0
         endif
          if(kk.eq.7) then
           Jz=0.9d0
          defectF=1.16d0
          alphaF=0.0d0
         endif
         if(kk.eq.8) then
           Jz=0.76d0
          defectF=0.76d0
          alphaF=0.0d0
         endif
         if(kk.eq.9) then
           Jz=0.76d0
          defectF=0.86d0
```

```
          alphaF=0.0d0
         endif
         if(kk.eq.10) then
           Jz=0.76d0
          defectF=0.9d0
          alphaF=0.0d0
         endif
         if(kk.eq.11) then
           Jz=0.76d0
          defectF=1.04d0
          alphaF=0.0d0
         endif
          if(kk.eq.12) then
           Jz=0.48d0
          defectF=0.00d0
          alphaF=0.76d0
         endif
          if(kk.eq.13) then
           Jz=0.48d0
          defectF=0.76d0
          alphaF=0.00d0
         endif

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! OUTPUT FILES
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      write(JzC,10) Jz
      write(alC,10) alphaF
      write(deC,10) defectF
10   format(F4.2)



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c MF-BASIS = LDOS = SF
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       allocate(work(7*dimTotal))
       allocate(UnpVecSite(dimTotal,dimTotal))
       allocate(UnpEig(dimTotal))
          call HamiltonianOfMF()
       CALL DSYEV('V','U',dimTotal,UnpVecSite,dimTotal,UnpEig,WORK,7*dimTotal,INFO)

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c HAMILTONIAN in the SITE-BASIS
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       allocate(PertVecSite(dimTotal,dimTotal))
       call HamiltonianSite()

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! Before diagonalizing the total Hamiltonian in the site basis: PertVecSite
! we can use it to get the equivalent matrix in the XXZ-basis
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

!c Multiplying matrices
       dd=dimTotal
```

```
       allocate(r1(dimTotal,dimTotal))
       allocate(HamMF(dimTotal,dimTotal))
       call
DGEMM('n','n',dd,dd,dd,1.0d0,PertVecSite,dd,UnpVecSite,dd,0.0d0,r1,dd)
       call DGEMM('t','n',dd,dd,dd,1.0d0,UnpVecSite,dd,r1,dd,0.0d0,HamMF,dd)
       deallocate(r1)

       allocate(XVALT(dimTotal))
       allocate(IMULT(dimTotal))
       do k=1,dimTotal
         XVALT(k)=HamMF(k,k)
       enddo

! Ordering the diagonal elements
       call MRGRNK (XVALT, IMULT)
       deallocate(XVALT)

       allocate(Haux(dimTotal,dimTotal))
       DO i=1,dimTotal
       DO j=1,dimTotal
         Haux(i,j) = HamMF(IMULT(i),IMULT(j))
       ENDDO
       ENDDO
       HamMF=Haux
       deallocate(Haux)
       deallocate(IMULT)

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!                 CONNECTIVITY of each line
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! Variance of |H(i,j)| for all off-elements (ABSOLUTE VALUE!!)
!
       saiCO='tConVoff_L'//LC//'u'//up//'JzF'//JzC//'dF'//deC//'Af'//alC//'db01.dat'
       OPEN(unit=50, FILE=saiCO,STATUS='UNKNOWN')
       write(50,*) '# DiagEn, connectivity, var , aveV/M ,(max-min)/M, off '
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                                square=0.0d0
                                ave=0.0d0
                                tot=0
                        Do i=1,dimTotal-1
                                Do j=i+1,dimTotal
                                        tot=tot+1
                                        square=square+HamMF(i,j)**2
                                        ave=ave+dabs(HamMF(i,j))
                                Enddo
                        ENDDO
                        ave=ave/dble(tot)
                        square=square/dble(tot)
                        var=square-ave**2
!test       write(*,*) var

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
       Do i=1,dimTotal
         connectMF=0
         mV=0.0d0
         maxi = - 1000000.0d0
         mini = 1000000.0d0
```

```
          offi=0.0d0

          DO j=1,dimTotal-i
           offi=offi+dabs(HamMF(j,j+i))
          ENDDO

          Do j=i+1,dimTotal
            if(dabs(HamMF(i,j)).gt.dabs(var)) then
            connectMF=connectMF+1
            mV=mV+dabs(HamMF(i,j))
              if(HamMF(j,j).gt.maxi) then
              maxi=HamMF(j,j)
              endif
              if(HamMF(j,j).lt.mini) then
              mini=HamMF(j,j)
              endif
            endif
          Enddo

          Do j=1,i-1
              if(dabs(HamMF(i,j)).gt.dabs(var)) then
            connectMF=connectMF+1
            mV=mV+dabs(HamMF(i,j))
              if(HamMF(j,j).gt.maxi) then
              maxi=HamMF(j,j)
              endif
              if(HamMF(j,j).lt.mini) then
              mini=HamMF(j,j)
              endif
            endif
          Enddo

          divi = dble(connectMF)

         offi=offi/dble(dimTotal-i)
         write(50,150) HamMF(i,i),connectMF,var,mV/divi,(maxi-mini)/divi,offi
        Enddo
150                     format(E16.7,i6,4E16.7)
        close(50)


! GET SPACE by DEALLOCATION
        deallocate(work)
        deallocate(PertVecSite)
        deallocate(UnpVecSite)
        deallocate(UnpEig)
        deallocate(HamMF)


! closing kk for the values of Jz and alpha
        ENDDO

! GET SPACE by DEALLOCATION
        deallocate(site_basis)

!ccccccccccccccccccccccccccccccccccccccccccccccccccc
!c END END END END END END END END END END END END
!ccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
!       STOP
        END




!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccc FUNCTION FUNCTION FUNCTION FUNCTION ccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

                        FUNCTION BINOMIAL(X,Y)

      implicit none
                        INTEGER (kind=4) :: BINOMIAL
                        INTEGER (kind=4) :: X,Y,k
                        INTEGER (kind=8) :: num,den

                        num=1
                        den=1

                        IF(Y.GT.(X-Y)) then
                                DO k= Y+1, X
                                        num=k*num
                                ENDDO
                                DO k= 1, (X-Y)
                                        den=k*den
                                ENDDO
                                BINOMIAL=num/den
                        ELSE
                                DO k= (X-Y)+1, X
                                        num=k*num
                                ENDDO
                                DO k= 1, Y
                                        den=k*den
                                ENDDO
                                BINOMIAL=num/den
                        ENDIF

                        return
                        END FUNCTION BINOMIAL
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



!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 that leads to MF-BASIS
!
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine HamiltonianOfMF()

      use variables
      implicit none

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


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


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

      Do i=1,dimTotal

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

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

!ccccccccccccccc NN ccccccccccccccccccccccccccc
        Do j=1,chain_size-1
          UnpVecSite(i,i)=UnpVecSite(i,i)+(Jz/4.d0)*(-1.0d0)**
(site_basis(i,j)+site_basis(i,j+1))
        enddo

!ccccccccccccccc NNN ccccccccccccccccccccccccccc
         Do j=1,chain_size-2
          UnpVecSite(i,i)=UnpVecSite(i,i)+alphaI*(Jz/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
                UnpVecSite(i,j)=UnpVecSite(i,j)+Jxy/2.0d0
                UnpVecSite(j,i)=UnpVecSite(j,i)+Jxy/2.0d0
              ENDIF
            enddo

!ccccccccccccccc NNN ccccccccccccccccccccc
            do k = 1, chain_size-2
              IF(bip(k)*bip(k+2).EQ.1) then
                UnpVecSite(i,j)=UnpVecSite(i,j)+alphaI*Jxy/2.0d0
                UnpVecSite(j,i)=UnpVecSite(j,i)+alphaI*Jxy/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 HAMILTONIAN that gives the MF-BASIS
      return
      end subroutine HamiltonianOfMF
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc







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

      subroutine HamiltonianSite()

      use variables
      implicit none

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


!c INITIALIZATION
      Do i=1,dimTotal
        Do j=1,dimTotal
```

```
          PertVecSite(i,j)=0.0d0
        Enddo
      Enddo


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

      Do i=1,dimTotal

!cccccccc DEFECT on SITE 1 ccccccccccccccc
        PertVecSite(i,i)=PertVecSite(i,i)+0.5d0*defborder*(-1.0d0)**(1+site_basis(i,1))
!cccccccc DEFECT on MIDDLE ccccccccccccccc
        PertVecSite(i,i)=PertVecSite(i,i)+0.5d0*defectF*(-1.0d0)**
(1+site_basis(i,chain_size/2))

!ccccccccccccccc NN ccccccccccccccccccccc
        Do j=1,chain_size-1
          PertVecSite(i,i)=PertVecSite(i,i)+(Jz/4.d0)*(-1.0d0)**
(site_basis(i,j)+site_basis(i,j+1))
        enddo

!ccccccccccccccc NNN ccccccccccccccccccccc
         Do j=1,chain_size-2
       PertVecSite(i,i)=PertVecSite(i,i)+alphaF*(Jz/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
                PertVecSite(i,j)=PertVecSite(i,j)+Jxy/2.0d0
                PertVecSite(j,i)=PertVecSite(j,i)+Jxy/2.0d0
              ENDIF
            enddo

!ccccccccccccccc NNN ccccccccccccccccccccc
            do k = 1, chain_size-2
              IF(bip(k)*bip(k+2).EQ.1) then
                PertVecSite(i,j)=PertVecSite(i,j)+alphaF*Jxy/2.0d0
```

```
                PertVecSite(j,i)=PertVecSite(j,i)+alphaF*Jxy/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 HAMILTONIAN
!  in the SITE-BASIS
      return
      end subroutine HamiltonianSite
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc




!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


       Subroutine mrgrnk (XDONT, IRNGT)
      use variables
      implicit none
! ____________________________________________________________
!   MRGRNK = Merge-sort ranking of an array
!   For performance reasons, the first 2 passes are taken
!   out of the standard loop, and use dedicated coding.
! ____________________________________________________________
! ____________________________________________________________
      Real (kind=8), Dimension (dimTotal), Intent (In) :: XDONT
      Integer (kind=4), Dimension (dimTotal), Intent (Out) :: IRNGT
! ____________________________________________________________
      Real (kind=8) :: XVALA, XVALB
!
      Integer (kind=4), Dimension (SIZE(IRNGT)) :: JWRKT
      Integer (kind=4) :: LMTNA, LMTNC, IRNG1, IRNG2
      Integer (kind=4) :: NVAL, IIND, IWRKD, IWRK, IWRKF, JINDA, IINDA, IINDB
!
      NVAL = Min (SIZE(XDONT), SIZE(IRNGT))
      Select Case (NVAL)
      Case (:0)
         Return
      Case (1)
         IRNGT (1) = 1
         Return
      Case Default
         Continue
      End Select
!
!  Fill-in the index array, creating ordered couples
!
      Do IIND = 2, NVAL, 2
```

```
         If (XDONT(IIND-1) <= XDONT(IIND)) Then
            IRNGT (IIND-1) = IIND - 1
            IRNGT (IIND) = IIND
         Else
            IRNGT (IIND-1) = IIND
            IRNGT (IIND) = IIND - 1
         End If
      End Do
      If (Modulo(NVAL, 2) /= 0) Then
         IRNGT (NVAL) = NVAL
      End If
!
!  We will now have ordered subsets A - B - A - B - ...
!  and merge A and B couples into     C   -   C   - ...
!
      LMTNA = 2
      LMTNC = 4
!
!  First iteration. The length of the ordered subsets goes from 2 to 4
!
      Do
         If (NVAL <= 2) Exit
!
!   Loop on merges of A and B into C
!
         Do IWRKD = 0, NVAL - 1, 4
            If ((IWRKD+4) > NVAL) Then
               If ((IWRKD+2) >= NVAL) Exit
!
!   1 2 3
!
               If (XDONT(IRNGT(IWRKD+2)) <= XDONT(IRNGT(IWRKD+3))) Exit
!
!   1 3 2
!
               If (XDONT(IRNGT(IWRKD+1)) <= XDONT(IRNGT(IWRKD+3))) Then
                  IRNG2 = IRNGT (IWRKD+2)
                  IRNGT (IWRKD+2) = IRNGT (IWRKD+3)
                  IRNGT (IWRKD+3) = IRNG2
!
!   3 1 2
!
               Else
                  IRNG1 = IRNGT (IWRKD+1)
                  IRNGT (IWRKD+1) = IRNGT (IWRKD+3)
                  IRNGT (IWRKD+3) = IRNGT (IWRKD+2)
                  IRNGT (IWRKD+2) = IRNG1
               End If
               Exit
            End If
!
!   1 2 3 4
!
            If (XDONT(IRNGT(IWRKD+2)) <= XDONT(IRNGT(IWRKD+3))) Cycle
!
!   1 3 x x
!
            If (XDONT(IRNGT(IWRKD+1)) <= XDONT(IRNGT(IWRKD+3))) Then
```

```
               IRNG2 = IRNGT (IWRKD+2)
               IRNGT (IWRKD+2) = IRNGT (IWRKD+3)
               If (XDONT(IRNG2) <= XDONT(IRNGT(IWRKD+4))) Then
!   1 3 2 4
                  IRNGT (IWRKD+3) = IRNG2
               Else
!   1 3 4 2
                  IRNGT (IWRKD+3) = IRNGT (IWRKD+4)
                  IRNGT (IWRKD+4) = IRNG2
               End If
!
!   3 x x x
!
            Else
               IRNG1 = IRNGT (IWRKD+1)
               IRNG2 = IRNGT (IWRKD+2)
               IRNGT (IWRKD+1) = IRNGT (IWRKD+3)
               If (XDONT(IRNG1) <= XDONT(IRNGT(IWRKD+4))) Then
                  IRNGT (IWRKD+2) = IRNG1
                  If (XDONT(IRNG2) <= XDONT(IRNGT(IWRKD+4))) Then
!   3 1 2 4
                     IRNGT (IWRKD+3) = IRNG2
                  Else
!   3 1 4 2
                     IRNGT (IWRKD+3) = IRNGT (IWRKD+4)
                     IRNGT (IWRKD+4) = IRNG2
                  End If
               Else
!   3 4 1 2
                  IRNGT (IWRKD+2) = IRNGT (IWRKD+4)
                  IRNGT (IWRKD+3) = IRNG1
                  IRNGT (IWRKD+4) = IRNG2
               End If
            End If
         End Do
!
!  The Cs become As and Bs
!
         LMTNA = 4
         Exit
      End Do
!
!  Iteration loop. Each time, the length of the ordered subsets
!  is doubled.
!
      Do
         If (LMTNA >= NVAL) Exit
         IWRKF = 0
         LMTNC = 2 * LMTNC
!
!   Loop on merges of A and B into C
!
         Do
            IWRK = IWRKF
            IWRKD = IWRKF + 1
            JINDA = IWRKF + LMTNA
            IWRKF = IWRKF + LMTNC
            If (IWRKF >= NVAL) Then
```

```
               If (JINDA >= NVAL) Exit
               IWRKF = NVAL
            End If
            IINDA = 1
            IINDB = JINDA + 1
!
!   Shortcut for the case when the max of A is smaller
!   than the min of B. This line may be activated when the
!   initial set is already close to sorted.
!
!          IF (XDONT(IRNGT(JINDA)) <= XDONT(IRNGT(IINDB))) CYCLE
!
!  One steps in the C subset, that we build in the final rank array
!
!  Make a copy of the rank array for the merge iteration
!
            JWRKT (1:LMTNA) = IRNGT (IWRKD:JINDA)
!
            XVALA = XDONT (JWRKT(IINDA))
            XVALB = XDONT (IRNGT(IINDB))
!
            Do
               IWRK = IWRK + 1
!
!  We still have unprocessed values in both A and B
!
               If (XVALA > XVALB) Then
                  IRNGT (IWRK) = IRNGT (IINDB)
                  IINDB = IINDB + 1
                  If (IINDB > IWRKF) Then
!  Only A still with unprocessed values
                     IRNGT (IWRK+1:IWRKF) = JWRKT (IINDA:LMTNA)
                     Exit
                  End If
                  XVALB = XDONT (IRNGT(IINDB))
               Else
                  IRNGT (IWRK) = JWRKT (IINDA)
                  IINDA = IINDA + 1
                  If (IINDA > LMTNA) Exit! Only B still with unprocessed values
                  XVALA = XDONT (JWRKT(IINDA))
               End If
!
            End Do
         End Do
!
!  The Cs become As and Bs
!
         LMTNA = 2 * LMTNA
      End Do
!
      Return
!
       End Subroutine mrgrnk
```