```
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! This code does the following uses as initial state a site-basis vector
!
! It evolves this state according to a final Hamiltonian and computes
! the evolution of the observables
!
!
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

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

       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

! PARAMETERS of the Hamiltonian
       real (kind=8) :: Jxy,Jz,defect,lambda

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

! Eigenvectors and eigenvalues
       real (kind=8), dimension(:,:), allocatable :: VecSite
       real (kind=8), dimension(:), allocatable :: Eig

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

! 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) :: aux,auxZ,auxX,PVScos,PVSsin
       real (kind=8) :: PVScos2,PVSsin2,PVScos3,PVSsin3
       INTEGER (kind=4), dimension(:), allocatable :: Npair,cxxI,cxxJ
       INTEGER (kind=4), dimension(:), allocatable :: cxxI2,cxxJ2,cxxI3,cxxJ3
       INTEGER (kind=4), dimension(:,:), allocatable :: pairI,pairJ

! CONSTANTS
```

```
       real (kind=8) :: pi


! TIME EVOLUTION
       real (kind=8), dimension(:), allocatable :: Initial
       real (kind=8) :: auxInt
       INTEGER (kind=4) :: tt,tinitial,tfinal,iniState
       real (kind=8) :: dt,time
       real (kind=8), dimension(:), allocatable :: Calpha
       real (kind=8), dimension(:), allocatable :: CosSite,SinSite
       real (kind=8), dimension(:), allocatable :: CosAlpha,SinAlpha

! For the OUTPUT files
       character(len=1) up,inC,iniSite
       character(len=2) LC
       character(len=3) def
       character(len=4) JzC,alC,Eni
       character(len=40) saiIEKE,saiStzz,saiStxx,saiMag,saiTIME
       character(len=40) saiCzz,saiCxx,saiEn

       end module


!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!             Program starts here
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       Program EvolveObservables

       use variables
       implicit none


!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! PARAMETERS
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       chain_size=16
       upspins=8
       dimTotal=12870

!       chain_size=14
!       upspins=7
!       dimTotal=3432

!       chain_size=12
!       upspins=6
!       dimTotal=984

!       chain_size=6
!       upspins=3
!       dimTotal=20
```

```
       dd=dimTotal

       Jxy=1.0d0
       defect=0.0d0

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

! TIME INTERVALE
       tinitial=0
       tfinal=70000
       dt=0.1d0

!ccccccccccccccccccccccccccccc
! INITIAL STATE
!ccccccccccccccccccccccccccccc
       iniState=1 ! this is the DOMAIN WALL: 11110000
!       iniSite = 'NS'
!       iniSite = 'PS'
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! Other INITIAL STATES from the SITE-BASIS VECTORS
!       Do i=1,dimTotal
!         auxInt=0.0d0
!!!!!!!!!! NS NS NS NS !!!!!!!!!!
!       IF(iniSite.eq.'NS') then
!         Do j=1,chain_size,2
!           auxInt=auxInt+basis(i,j)
!         Enddo
!         If(auxInt.eq.chain/2) then
!           iniState=i
!         endif
!       ENDIF
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!! PS PS PS PS !!!!!!!!!!
!       IF(iniSite.eq.'PS') then
!         Do j=2,chain_size,4
!           auxInt=auxInt+basis(i,j)
!         Enddo
!         Do j=3,chain,4
!           auxInt=auxInt+basis(i,j)
!         Enddo
!         If(auxInt.eq.chain/2) then
!           iniState=i
!         endif
!       ENDIF
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!       Enddo
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      write(inC,12) iniState
12    format(i1.1)
      write(LC,8) chain_size
```

```
8     format(i2.2)
       write(up,9) upspins
9     format(i1)
       write(def,10) defect
10    format(F3.1)


!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c CREATE SITE-BASIS
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       allocate(basis(dimTotal,chain_size))
       call SiteBasis()

! ALLOCATION needed during the whole CODE

       allocate(VecSite(dimTotal,dimTotal))
       allocate(Eig(dimTotal))

       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))

       allocate(Calpha(dimTotal))
       allocate(CosSite(dimTotal))
       allocate(SinSite(dimTotal))
       allocate(CosAlpha(dimTotal))
       allocate(SinAlpha(dimTotal))



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c DO-LOOP for the values of Jz and lambda
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
        Do kk=1,6
          if(kk.eq.1) then
            Jz=1.0d0
            lambda=0.0d0
          endif
          if(kk.eq.2) then
            Jz=0.5d0
            lambda=0.0d0
          endif
          if(kk.eq.3) then
            Jz=1.0d0
            lambda=0.4d0
          endif
          if(kk.eq.4) then
            Jz=0.5d0
            lambda=0.4d0
          endif
          if(kk.eq.5) then
            Jz=1.0d0
            lambda=1.0d0
          endif
          if(kk.eq.6) then
            Jz=0.5d0
            lambda=1.0d0
          endif



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! OUTPUT FILES
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       write(JzC,11) Jz
       write(alC,11) lambda
11    format(F4.2)


! INTERACTION ENERGY and KINETIC ENERGY   ccccccccccccccccccc
        saiIEKE='IEKE_L'//LC//'u'//up//'A'//alC//'Jz'//JzC//'def'//def//'i'//inC//'.dat'
        OPEN(unit=20, FILE=saiIEkE,STATUS='UNKNOWN')
!  SITE MAGNETIZATION ccccccccccccccccccccccccccccccccccccccccc
        saiMag='Mag_L'//LC//'u'//up//'A'//alC//'Jz'//JzC//'def'//def//'i'//inC//'.dat'
        OPEN(unit=25, FILE=saiMag,STATUS='UNKNOWN')
!  SPIN-SPIN STRUCTURE FACTOR ccccccccccccccccccccccccccccccc
        saiStzz='Stzz_L'//LC//'u'//up//'A'//alC//'Jz'//JzC//'def'//def//'i'//inC//'.dat'
        OPEN(unit=30, FILE=saiStzz,STATUS='UNKNOWN')
        saiStxx='Stxx_L'//LC//'u'//up//'A'//alC//'Jz'//JzC//'def'//def//'i'//inC//'.dat'
        OPEN(unit=40, FILE=saiStxx,STATUS='UNKNOWN')
!  SPIN-SPIN CORRELATION sites L/2,/2+1 ccccccccccccccccccccccccccccc

saiCzz='CzzMid123_L'//LC//'u'//up//'A'//alC//'Jz'//JzC//'def'//def//'i'//inC//'.dat'
        OPEN(unit=50, FILE=saiCzz,STATUS='UNKNOWN')

saiCxx='CxxMid123_L'//LC//'u'//up//'A'//alC//'Jz'//JzC//'def'//def//'i'//inC//'.dat'
```

```
       OPEN(unit=51, FILE=saiCxx,STATUS='UNKNOWN')
!  ENERGIES ccccccccccccccccccccccccccccc
       saiEn='Energy_L'//LC//'u'//up//'A'//alC//'Jz'//JzC//'def'//def//'i'//inC//'.dat'
       OPEN(unit=55, FILE=saiEn,STATUS='UNKNOWN')


!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c TOTAL HAMILTONIAN in the SITE-BASIS
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       allocate(work(7*dimTotal))
       call HamiltonianFinal()

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c EIGENVALUES and EIGENSTATES in the SITE-BASIS
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       CALL DSYEV('V','U',dimTotal,VecSite,dimTotal,Eig,WORK,7*dimTotal,INFO)
       deallocate(work)



!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c INITIAL STATE = SITE-BASIS vector
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       allocate(Initial(dimTotal))
        Do i=1,dimTotal
         Initial(i)=0.0
         write(55,56) i, Eig(i)
        Enddo
        Initial(iniState)=1.0d0

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! TIME DEPENDENCE of the INITIAL STATE
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

! write the initial state in the basis of eigenstates of the Total Hamiltonian

       call DGEMV('t',dd,dd,1.0d0,VecSite,dd,Initial,1,0.0d0,Calpha,1)
       deallocate(Initial)

! get the time dependence: cos(E.t) - i sin(E.t)
! TIME-LOOP

       DO tt=tinitial,tfinal
         time=dble(tt)*dt
         Do i=1,dimTotal
           CosAlpha(i)=Calpha(i)*dcos( time*Eig(i) )
           SinAlpha(i)= - Calpha(i)*dsin( time*Eig(i) )
         Enddo
!
! now bring it back to the SITE-BASIS to compute the observables
!
```

```
          call DGEMV('n',dd,dd,1.0d0,VecSite,dd,CosAlpha,1,0.0d0,CosSite,1)
          call DGEMV('n',dd,dd,1.0d0,VecSite,dd,SinAlpha,1,0.0d0,SinSite,1)


!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!             OBSERVABLES inside the TIME-LOOP
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

        call ObservablesTime()

! OUTPUT DATA
        write(20,120) time, IE, KE
        write(25,125) time, (SiteMag(j),j=1,chain_size)
        write(30,130) time, (Stzz(jk), jk=0,chain_size/2)
        write(40,130) time, (Stxx(jk), jk=0,chain_size/2)
        write(50,121) time, Czz,Czz2,Czz3
        write(51,121) time, Cxx,Cxx2,Cxx3
!


! end the TIME-LOOP
        ENDDO

! FORMAT
120     format(F8.1,2E18.9)
121     format(F8.1,3E18.9)
! aqui: CAREFUL with dimensions (just use a big one)
125     format(F8.1,20E18.9)
! aqui: CAREFUL with dimensions
130     format(F8.1,20E18.9)
56     format(I6,E18.9)

        close(20)
        close(25)
        close(30)
        close(40)
        close(50)
        close(55)

! closing kk for the values of Jz and lambda
        ENDDO


! DEALLOCATION
        deallocate(basis)

        deallocate(VecSite)
        deallocate(Eig)

        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)

        deallocate(Calpha)
        deallocate(CosSite)
        deallocate(SinSite)
        deallocate(CosAlpha)
        deallocate(SinAlpha)

!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
          basis(ib,jb)=0
        enddo
      enddo

      ii=1

71    call nexksb(chain_size,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
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc




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

```
! and also to get the elements of the operators IE and KE
!
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      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
          VecSite(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
        VecSite(i,i)=VecSite(i,i)+0.5d0*defect*(-1.0d0)**(1+basis(i,1))
! 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)**(basis(i,mid)+basis(i,mid+1))
        Czza2(i)=Czza2(i)+(0.25d0)*(-1.0d0)**(basis(i,mid)+basis(i,mid+2))
        Czza3(i)=Czza3(i)+(0.25d0)*(-1.0d0)**(basis(i,mid)+basis(i,mid+3))

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

!ccccccccccccccc NNN ccccccccccccccccccccc
         Do j=1,chain_size-2
```

```
       VecSite(i,i)=VecSite(i,i)+lambda*(Jz/4.d0)*(-1.0d0)**(basis(i,j)+basis(i,j+2))
       IEdiag(i)=IEdiag(i)+lambda*(Jz/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_size
            bip(k) = mod(basis(i,k) + 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
                VecSite(i,j)=VecSite(i,j)+Jxy/2.
                VecSite(j,i)=VecSite(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
                VecSite(i,j)=VecSite(i,j)+lambda*Jxy/2.
                VecSite(j,i)=VecSite(j,i)+lambda*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
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!
! SUBROUTINE to compute the OBSERVABLES and time t
!
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine ObservablesTime()

      use variables
      implicit none



! INITIALIZATION 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

! IE, SITE MAGNETIZATION, and Stzz !!!!!!!!!!!!!!!!!!!!!!!!!
        DO i=1, dimTotal
          aux=CosSite(i)**2 + SinSite(i)**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+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)**(basis(i,jj)+basis(i,jj+j))
                           enddo
                           ZZ(k)=ZZ(k)+phase*auxZ
            Enddo
            Stzz(k)=Stzz(k)+aux*ZZ(k)
          Enddo
        ENDDO
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!Cxx  (L/2,L/2+1)
        Do j=1,nxx
         PVScos=CosSite(cxxI(j))*CosSite(cxxJ(j))
         PVSsin=SinSite(cxxI(j))*SinSite(cxxJ(j))
         Cxx=Cxx+0.5*(PVScos+PVSsin)
        Enddo
!Cxx  (L/2,L/2+2)
        Do j=1,nxx2
```

```
        PVScos2=CosSite(cxxI2(j))*CosSite(cxxJ2(j))
        PVSsin2=SinSite(cxxI2(j))*SinSite(cxxJ2(j))
        Cxx2=Cxx2+0.5*(PVScos2+PVSsin2)
       Enddo
!Cxx  (L/2,L/2+3)
       Do j=1,nxx3
        PVScos3=CosSite(cxxI3(j))*CosSite(cxxJ3(j))
        PVSsin3=SinSite(cxxI3(j))*SinSite(cxxJ3(j))
        Cxx3=Cxx3+0.5*(PVScos3+PVSsin3)
       Enddo
! KE
! NOTE: it is 2(PVScos+PVSsin)(Jxy/2), so the 2's cancel
       Do j=1,Npair(1)
         PVScos=CosSite(pairI(1,j))*CosSite(pairJ(1,j))
         PVSsin=SinSite(pairI(1,j))*SinSite(pairJ(1,j))
         KE=KE+(PVScos+PVSsin)
       Enddo
       Do j=1,Npair(2)
         PVScos=CosSite(pairI(2,j))*CosSite(pairJ(2,j))
         PVSsin=SinSite(pairI(2,j))*SinSite(pairJ(2,j))
         KE=KE+lambda*(PVScos+PVSsin)
       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)
         PVScos=CosSite(pairI(j,jj))*CosSite(pairJ(j,jj))
         PVSsin=SinSite(pairI(j,jj))*SinSite(pairJ(j,jj))
         auxX=auxX+2.0d0*(PVScos+PVSsin)
           Enddo
           Stxx(k)=Stxx(k)+phase*auxX
         Enddo
       Enddo

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

!  END of SUBROUTINE that computes the evolved observables
!  using the SITE-BASIS
      return
      end subroutine ObservablesTime
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```