Anexo 4: Programa Fortran para el calculo de transporte electronico

```
module INIDATA
use UTILITIES
!-------------------------------------------------------------------------
---------
!Purpose: This module load the initial input paraeters and the initial
data
!needed for transport calculations. The parameter are divide into the
system
!parameters and electronic parameters respectivelly. This module use
the
!Tag's [1] suroutines defined in the module UTILITIES in order to
facilitate
!the redin of the input parameters, and also external LAPACK [2]
subroutines
!for solve the inversion and eigen proplem of a matrix.
!
!(i)the system parameters: the name of the project (DNA or
PROTEIN),the number of
!levels (NL) for the reduce matrix, the number of monomers (NM),i.e.
number of
!nucleotides or amino acids in the sequence,the number of actual pair
(NAM) in the sequence,
!the zise (LH) of the Hamiltonian matrix of each pair,the HOMO's
position (HOMO1,HOMO2)
! in the Fock matrix of the fragment the length of the system
(NS),i.e. the number of
!different combination of the monomers;
!(ii)Electronic parameters: elctronic temperature (Tp),the lead
(Lead), the Fermi level
!of the lead (Ef),the chemical potentials (MuL,MuR),the scape rate
(Gm),and the DC
!applied voltage (DC_volt)
!
!The data for he system is stored inthe follows array:
!a)the 1D SeqM, CouplM and ActPairM arrays stored the sequencen, the
coupling and the
!number of actual pair of monomers.

!b)the 1D arrays ListSyst and ListOrd stored in a list the monomers
members of the project,
!and the number and defined order in which the data is stored

!c)the 1D arrays ListLH, ListHOM1 and ListHOM2; stored the zise of the
Fock matirx and the
!position in the Fock matrix of the HOMO's of fragments for each pairs
of monomers(*)

!d)the array ListLev, accordin with the number of level NL and the
HOMO's position, stored
```

```
!the set of levels of each fragment to be used in the construction of
the reduce matrix,
!i.e. stored ......HOMO1-1,HOMO1,LUMO1,......HOMO2-1,HOMO2,LUMO2, of
each pair(*)
!
!e)the arrays ListH0,ListS and ListH; stored the isolated Hamiltonian
(of fragments)matrix,
!the overlap matrix and the coupling Hamiltonian (fragments + ending
groups) matirx of each
!pair of monomers(*)
!
!f)The arrays ListH0Red and ListSRed, stored the reduce isolated
hamitonian of fragments and
!overlap matrix for ecah pairs according to the set of levels stored
in ListLev(*)
!
!g)the arrays ListEva and ListEve, stored the eigenvalues and
eigenvectors of the Hamiltonian
!matrix (generalize eigen problem) of each pair(*)
!
!h)The arrays ListEveRed stored the reduce set of eigenvector for ecah
pairs, according to the
!set of levels stored in ListLev(*)
!
!Finally a set of constant needed for calculations are defined: the
value of Pi, the electon
!charge(Ech), the Plank constant(Hp),the Boltzman constant(Kb), and a
conversion factor(CF)(**)
!
!
!Note:
!(*)All stored follows the order defined in ListOrd
!(**)In Hartree units
!NMAX define the maximum zise allocation for the H0,S,H, Eve,Eva,
matrices, since pairs all
!different in zise
!H0 is a temporal array for intermediate calculations of isolated
Hamiltonian
!
! Record of revisions
!   Date      Programer          Descripion of change
! -------   ---------------   ---------------------------------------
! 30/01/2023   J. R. Alvarez    Original code
!---------------------------------------------------------------------
----------------------------------


implicit none
integer                                 ::NMAX=1500
integer                                 ::NL
integer                                 ::NM
integer                                 ::NC
```

```
integer                                     ::Cicle
integer                                     ::LHX
integer                                     ::LHXY
integer                                     ::HOMOX
integer                                     ::HOMOXY1
integer                                     ::HOMOXY2
integer                                     ::NS
integer                                     ::NB
real*8                                      ::Tp
real*8                                      ::Ef
real*8                                      ::DC_volt
real*8                                      ::Gm
real*8                                      ::MuL
real*8                                      ::MuR
character*2                                 ::Lead
character*10                                ::Project
character(5)                                ::afileN
character*100                               ::path
integer                                     ::IntSub
integer                                     ::ifileN
!---------------------------------------------------------------------
--
!multidimensinal array for stored data needed for future calculations
!---------------------------------------------------------------------
--
character(1),allocatable,dimension(:)  ::Seq
character(1),allocatable,dimension(:,:)::SeqM
character(2),allocatable,dimension(:)  ::Coupl
character(1),allocatable,dimension(:)  ::ListOrdX
character(2),allocatable,dimension(:)  ::ListOrdXY
integer,allocatable,dimension(:)       ::ListLHX
integer,allocatable,dimension(:)       ::ListLHXY
integer,allocatable,dimension(:)       ::ListHOMX
integer,allocatable,dimension(:)       ::ListHOMXY1
integer,allocatable,dimension(:)       ::ListHOMXY2
real*8,allocatable,dimension(:,:)      ::totalS
real*8,allocatable,dimension(:,:)      ::totalH
real*8,allocatable,dimension(:,:,:)    ::rdVNSX
real*8,allocatable,dimension(:,:,:)    ::rdVNSXY
integer,allocatable,dimension(:,:)     ::ListLevX
integer,allocatable,dimension(:,:)     ::ListLevXY
real*8,allocatable,dimension(:,:,:)    ::ListSX
real*8,allocatable,dimension(:,:,:)    ::ListHX
real*8,allocatable,dimension(:,:,:)    ::ListSXY
real*8,allocatable,dimension(:,:,:)    ::ListHXY
real*8,allocatable,dimension(:,:,:)    ::ListSRedX
real*8,allocatable,dimension(:,:,:)    ::ListSRedXY
real*8,allocatable,dimension(:,:,:)    ::ListEveX
real*8,allocatable,dimension(:,:,:)    ::ListEvaX
real*8,allocatable,dimension(:,:,:)    ::ListEveRedX
real*8,allocatable,dimension(:,:,:)    ::ListEveXY
real*8,allocatable,dimension(:,:,:)    ::ListEvaXY
```

```
real*8,allocatable,dimension(:,:,:)    ::ListEveRedXY
!-------------------------------------------------------------------------
-----
!Parameters
!-------------------------------------------------------------------------
-----
real*8,parameter                       ::Pi       =
3.14159265358979323846d00
real*8,parameter                       ::Ech      = 1.0d0
real*8,parameter                       ::Hp       = 1.0d0
real*8,parameter                       ::Kb       = 8.6173324d-05
real*8,parameter                       ::CF       = 27.2116d00
real*8,parameter                       ::delta    = 1.0d-12
real*8,parameter                       ::test_E   = -5.50d00
contains

!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

subroutine input_data
!-------------------------------------------------------------------------
----------
!This subroutine load the initial data, thta define the system
parameters and
!the electronic parameters. This data is stored in the input_data.dat
file
!-------------------------------------------------------------------------
----------
integer                                 ::i,j,k,stat
character(100)                          ::inidata
character(100)                          ::seqfile
character(5)                            ::afileN
integer                                 ::ifileN
read*, ifileN

call chart_int(ifileN,afileN)
inidata= 'input_data-'//trim(adjustl(afileN))//'.dat'
open(UNIT=10,file=trim(inidata),status='old',action='read',IOSTAT=stat
)
if(stat /= 0)stop "Main: error opening ini file"
call FindTagA(10, "System =", Project)
call FindTagA(10, "Electrode =", Lead)
call FindTagI(10, "Number of levels =", NL)
call FindTagI(10, "Number of monomers =", NM)
call FindTagI(10, "Number of cicles =", NC)
call FindTagR(10, "DC bias voltage =", DC_volt)
call FindTagR(10, "Scape rate =", Gm)
call FindTagR(10, "Temperature =", Tp)
close(unit=10)

!-------------------------------------------------------------------------
------
```

```
!define the zise of the system, i.e. the number of different pairs
!-------------------------------------------------------------------
------

if(Project=='DNA')then
    NB=4
    allocate (ListOrdX(NB))
    ListOrdX=(/'A','C','G','T'/)
    path='/work/jacko0713/dna/dna-ab-data/'
    seqfile=trim(path)//'DNA_seq.dat'

  else if(Project=='RNA')then
    NB=4
    allocate (ListOrdX(NB))
    ListOrdX=(/'A','C','G','U'/)
    path='/work/jacko0713/dna/dna-ab-data/'
    seqfile=trim(path)//'RNA_seq.dat'

  else if(Project=='DNAM')then
    NB=5
    allocate (ListOrdX(NB))
    ListOrdX=(/'A','C','G','T','D'/)
    path='/work/jacko0713/dna/dna-ab-data/'
    seqfile=trim(path)//'DNAM_seq.dat'

  else if(Project=='DNAHM')then
    NB=6
    allocate (ListOrdX(NB))
    ListOrdX=(/'A','C','G','T','D','E'/)
    path='/work/jacko0713/dna/dna-ab-data/'
    seqfile=trim(path)//'DNAHM_seq.dat'

   else if(Project=='PROTEIN')then
    NB=20
    allocate (ListOrdX(NB))
    ListOrdX=(/'A','C','G','T','D','E'/)
    path='/work/jacko0713/dna/dna-ab-data/'
    seqfile=trim(path)//'PROTEIN_seq.dat'

   else
      write(*,*)'---PROJECT UNKNOWN--'
      stop

end if

!-------------------------------------------------------------------
----
!list with the pairs of monomers acording to the project
!-------------------------------------------------------------------
----

NS= (NB*(NB+1))/2
```

```
allocate (ListOrdXY(NS))

i=0
 do j=1,NB
  do k=j, NB
  i=i+1
   ListOrdXY(i)=ListOrdX(j)//ListOrdX(k)
  end do
 end do

!-----------------------------------------------------------------
-----
!with the type of electrode, define the Fermi level and the chemical
!-----------------------------------------------------------------
-----
if (Lead=='Al')then
 Ef = -4.250d00
  else
    if(Lead == 'Au')then
     Ef = -5.220d00
     else
       if(Lead == 'Pt')then
         Ef = -5.700d00
      else
        if(Lead == 'Gr')then
         Ef = -4.500d00
        else
         Write(*,*)'---ELECTRODE UNKNOWN---:'
         stop
      end if
    end if
  end if
end if

!**************Chemical potential*********************************
MuL=Ef-(DC_volt/2.0d00)
MuR=Ef+(DC_volt/2.0d00)

allocate(SeqM(NC,NM))
allocate(Seq(NM))
allocate(Coupl(NM))

open(UNIT=11,file=trim(adjustl(seqfile)),status='old',action='read',IO
STAT=stat)
if(stat /= 0)then
write(*,*) '---ERROR OPENING THE SEQFILE---'
stop
end if

do i=1,NC
 read(11,*)SeqM(i,:)
```

```
end do

close(unit=11)

Seq = SeqM(ifileN,:)

do i=1,NM-1
Coupl(i)=Seq(i)//Seq(i+1)
end do

Coupl(NM)=Seq(NM)//Seq(1)
return
end subroutine input_data

!*********************************************************************
***********
!
!
!*********************************************************************
**********

subroutine nucl_data
!---------------------------------------------------------------------

!---------------------------------------------------------------------

integer                                 ::k,m,n,p,j
character(150)                          ::ordermon,monomers_ab_data
character(1)                            ::x1
integer                                 ::ind1,frag_n1,occp1
real*8                                  ::energy1
character(1)                            ::frag_t1
integer                                 ::stat

!all input_data


!---------------------------------------------------------------------
---
!allocate the memory for the arrays that stored the data
!---------------------------------------------------------------------
---
allocate(ListOrdX(NB))
allocate(ListHOMX(NB))
allocate(ListLHX(NB))
allocate(ListSX(NB,NMAX,NMAX))
allocate(ListHX(NB,NMAX,NMAX))
allocate(ListLevX(NB,NL))
allocate(ListSRedX(NB,NL,NL))
allocate(ListEveX(NB,NMAX,NMAX))
allocate(ListEvaX(NB,NMAX,1))
allocate(ListEveRedX(NB,NL,NMAX))
```

```
!-----------------------------------------------------------------
-----------
!open the files _lev.dat, _ham.dat and _ovlp.dat, to read the
information of
!fragment and the Fock and overlap matix of each pair.
!-----------------------------------------------------------------
-----------

do m=1,NB

monomers_ab_data=trim(path)//ListOrdX(m)

open(UNIT=13,file=trim(monomers_ab_data)//'_lev.dat',status='old',acti
on='read',IOSTAT=stat)
if(stat /= 0)then
write(*,*) '---ERROR OPENING THE DATA OF MON LEVELS---'
stop
end if
!read the zise of the monomer Hamiltonian
  read(13,*,IOSTAT=stat)LHX

 ListLHX(m)=LHX

!read the zise of the index of levels, the occupations, energy of the
level,
!the fragment type and the fragment number

do k=1,LHX
   read(13,*,IOSTAT=stat)ind1,occp1,energy1,frag_t1,frag_n1

!stored the isolated hamiltonian
!    ListH0X(m,k,1)= energy1

!search for the HOMO's of each fragment
     if(occp1==2 .and. frag_t1==ListOrdX(m))then
       HOMOX=k
     end if
 end do
 close(unit=13)

   ListHOMX(m)=HOMOX


!-----------------------------------------------------------------
!read the Hamitonian and overlap amtrix
!-----------------------------------------------------------------
```

```
open(unit=14,file=trim(monomers_ab_data)//'_ham.dat',status='old',acti
on='read',IOSTAT=stat)
  if(stat /= 0)then
write(*,*) '---ERROR OPENING THE DATA OF MON HAMAMILTONIAN---'
stop
end if

open(unit=15,file=trim(monomers_ab_data)//'_ovlp.dat',status='old',act
ion='read',IOSTAT=stat)
  if(stat /= 0)then
write(*,*) '---ERROR OPENING THE DATA OF MON OVERLAP---'
stop
end if

   do n=1,LHX
      read(14,*)(ListHX(m,n,p),p=1,LHX)
      read(15,*)(ListSX(m,n,p),p=1,LHX)
   end do

   close(unit=14)
   close(unit=15)
end do

!---------------------------------------------------------------------
!account for the set of reduce levels, the reduce isolate Hamiltonian
!and overlap matrix neede for calculations of the reduce pertubation
!matrix and the total isolate GF of a chain of monomers
!---------------------------------------------------------------------

do m=1,NB
 if (NL==1)then
  ListLevX(m,1)=ListHOMX(m)-1
 else
  do n=1,NL-1    !only HOMO's
   ListLevX(m,n)=ListHOMX(m)-NL+n

!  ListLev(m,n)=ListHOM1(m)-NL+n+1
!  ListLev(m,n+NL)=ListHOM2(m)-NL+n+1
  end do
!LUMO+2 (r)
ListLevX(m,NL)=ListHOMX(m)+2

 end if
end do


do m=1,NB
  do n=1,NL
    do p=1,NL
     ListSRedX(m,n,p)=ListSX(m,ListLevX(m,n),ListLevX(m,p))
    end do
```

```
 end do
end do

return
end subroutine nucl_data

!*********************************************************************
**********************

subroutine pairs_data
!---------------------------------------------------------------------
!This subroutine open the files that contein all information about
!first principal calculations of the pair of monomomers, i.e. the
!information of the fragmnets,the Fock and overlap matrix. These
informaton
!are stored in following files calling A1A2_lev.dat, A1A2_ham.dat, and
!A1A2_ovlp.dat, where A1 and A2 are the monomers units according to
the
!project defined in the input data (DNA or PROTEIN).
!
!The file _lev.dat,containing iformation about levels position in the
!Fock-matrix of a pair, the total number of levels,the position of the
HOMO's
!of each fragment in the pair, and the isolate Hamiltonian matrix
(H0);
!whereas the files _ham.dat and _ovlp.dat just contain the Hamiltonian
and
!overlap matrix of the pair of monomers.
!
!
!
!This subroutine take as input the name of pair of monomers (file),
!and return as output the zise of Fock matrix (LH),the HOMO's position
!of each fragment (HOMO1 and HOMO2), and the isolated hamiltonian of
!pair (H0).
!---------------------------------------------------------------------

integer                                  ::k,m,n,p,j
character(150)                          ::dimers_ab_data
character(2)                             ::x,y
integer                                  ::ind2,frag_n2,occp2
real*8                                   ::energy2
character(2)                             ::frag_t2
integer                                  ::stat

!all input_data


!---------------------------------------------------------------------
---
!allocate the memory for the arrays that stored the data
```

```
!-----------------------------------------------------------------------
---
allocate(ListOrdXY(NS))
allocate(ListHOMXY1(NS))
allocate(ListHOMXY2(NS))
allocate(ListLHXY(NS))
allocate(ListSXY(NS,NMAX,NMAX))
allocate(ListHXY(NS,NMAX,NMAX))
allocate(ListLevXY(NS,2*NL))
allocate(ListSRedXY(NS,2*NL,2*NL))
allocate(ListEveXY(NS,NMAX,NMAX))
allocate(ListEvaXY(NS,NMAX,1))
allocate(ListEveRedXY(NS,2*NL,NMAX))


!-----------------------------------------------------------------------
-----------
!open the files _lev.dat, _ham.dat and _ovlp.dat, to read the
information of
!fragment and the Fock and overlap matix of each pair.
!-----------------------------------------------------------------------
-----------

do m=1,NS

 dimers_ab_data=trim(path)//ListOrdXY(m)

open(UNIT=17,file=trim(dimers_ab_data)//'_lev.dat',status='old',action
='read',IOSTAT=stat)
  if(stat /= 0)then
write(*,*) '---ERROR OPENING THE DATA OF DIMERS LEVELS---'
stop
end if

!read the zise of the Hamiltonian
  read(17,*,IOSTAT=stat)LHXY

 ListLHXY(m)=LHXY

 do k=1,LHXY
   read(17,*,IOSTAT=stat)ind2,occp2,energy2,frag_t2,frag_n2

!stored the isolated hamiltonian
!     ListH0XY(m,k,1)= energy2

!search for the HOMO's of each fragment
     if(occp2==2 .and. frag_t2==ListOrdXY(m)(1:1)//'1')then
       HOMOXY1=k
     end if

    if(occp2==2 .and. frag_t2==ListOrdXY(m)(2:2)//'2')then
       HOMOXY2=k
```

```
    end if

 end do
 close(unit=17)

   ListHOMXY1(m)=HOMOXY1
   ListHOMXY2(m)=HOMOXY2

!---------------------------------------------------------------------
!read the Hamitonian and overlap amtrix
!---------------------------------------------------------------------

open(unit=18,file=trim(dimers_ab_data)//'_ham.dat',status='old',action
='read',IOSTAT=stat)
    if(stat /= 0)then
write(*,*) '---ERROR OPENING THE DATA OF DIMERS HAMILTONIAN---'
stop
end if

open(unit=19,file=trim(dimers_ab_data)//'_ovlp.dat',status='old',actio
n='read',IOSTAT=stat)
    if(stat /= 0)then
write(*,*) '---ERROR OPENING THE DATA OF DIMERS OVERLAP---'
stop
end if

   do n=1,LHXY
      read(18,*)(ListHXY(m,n,p),p=1,LHXY)
      read(19,*)(ListSXY(m,n,p),p=1,LHXY)
    end do

   close(unit=18)
   close(unit=18)
end do

!---------------------------------------------------------------------
!account for the set of reduce levels, the reduce isolate Hamiltonian
!and overlap matrix neede for calculations of the reduce pertubation
!matrix and the total isolate GF of a chain of monomers
!---------------------------------------------------------------------
do m=1,NS
if (NL==1)then
 ListLevXY(m,1)=ListHOMXY1(m)-1
 ListLevXY(m,2)=ListHOMXY2(m)-1
else
 do n=1,NL-1   !only HOMO's
  ListLevXY(m,n)=ListHOMXY1(m)-NL+n
  ListLevXY(m,n+NL)=ListHOMXY2(m)-NL+n

 end do

ListLevXY(m,NL)=ListHOMXY1(m)+2
```

```
ListLevXY(m,2*NL)=ListHOMXY2(m)+2
end if
end do



do m=1,NS
  do n=1,2*NL
    do p=1,2*NL
     ListSRedXY(m,n,p)=ListSXY(m,ListLevXY(m,n),ListLevXY(m,p))
    end do
 end do
end do

return
end subroutine pairs_data

!*****************************************************************
************

subroutine nucl_coeff(nucl)
implicit none
integer                                    ::nucl
real*8,allocatable,dimension(:)            ::WORK1
integer                                    ::n,ii,jj,kmax
integer                                    ::info1,LWORK1,LWMAX1
character                                  ::v,u,l
real*8,allocatable,dimension(:,:)          ::tempEveX
real*8,allocatable,dimension(:)            ::tempEvaX
real*8,allocatable,dimension(:,:)          ::tempEveRX
real*8,allocatable,dimension(:,:)          ::tempSX

external dsygv


kmax=ListLHX(nucl)

LWORK1=max(1,3*kmax-1)

allocate(tempEvaX(kmax))
allocate(tempEveX(kmax,kmax))
allocate(tempEveRX(kmax,NL))
allocate(tempSX(kmax,kmax))
allocate(WORK1(LWORK1))

tempSX = 0.0d00
tempEveX = 0.0d0

do ii=1,kmax
 do jj=1,ii
  tempSX(ii,jj) = ListSX(nucl,ii,jj)
   tempEveX(ii,jj) = CF*ListHX(nucl,ii,jj)
```

```
 end do
end do

call
dsygv(1,'v','l',kmax,tempEveX,kmax,tempSX,kmax,tempEvaX,WORK1,LWORK1,i
nfo1)
  if (info1 .gt.0) then
    print *,'ERROR:,failed to compute evalues and evectors of
matrixH=',nucl
     stop
   end if


do ii=1,kmax
  ListEvaX(nucl,ii,1) = tempEvaX(ii)
 do jj=1,kmax
    ListEveX(nucl,ii,jj) = tempEveX(ii,jj)
 end do
end do

do ii = 1,NL
   do jj=1,kmax
     ListEveRedX(nucl,ii,jj) = ListEveX(nucl,ListLevX(nucl,ii),jj)
   end do
end do


!deallocate(tempEva(kmax))
!deallocate(tempEve(kmax,kmax))
!deallocate(tempEveR(kmax,2*NL))
!deallocate(tempS(kmax,kmax))
!deallocate(WORK(LWORK))

return
end subroutine nucl_coeff

!*****************************************************************
**********


subroutine pair_coeff(pnucl)
implicit none
integer                                    ::pnucl
real*8,allocatable,dimension(:)            ::WORK2
integer                                    ::n,ii,jj,kmax
integer                                    ::info2,LWORK2,LWMAX2
character                                  ::v,u,l
real*8,allocatable,dimension(:,:)          ::tempEveXY
real*8,allocatable,dimension(:)            ::tempEvaXY
real*8,allocatable,dimension(:,:)          ::tempEveRXY
real*8,allocatable,dimension(:,:)          ::tempSXY
```

```
external dsygv


kmax=ListLHXY(pnucl)

LWORK2=max(1,3*kmax-1)

allocate(tempEvaXY(kmax))
allocate(tempEveXY(kmax,kmax))
allocate(tempEveRXY(kmax,2*NL))
allocate(tempSXY(kmax,kmax))
allocate(WORK2(LWORK2))

tempSXY = 0.0d00
tempEveXY = 0.0d0

do ii=1,kmax
 do jj=1,ii
  tempSXY(ii,jj) = ListSXY(pnucl,ii,jj)
   tempEveXY(ii,jj) = ListHXY(pnucl,ii,jj)
 end do
end do

call
dsygv(1,'v','l',kmax,tempEveXY,kmax,tempSXY,kmax,tempEvaXY,WORK2,LWORK
2,info2)
  if (info2 .gt.0) then
    print *,'ERROR:,failed to compute evalues and evectors of
matrixH=',pnucl
     stop
   end if


do ii=1,kmax
  ListEvaXY(pnucl,ii,1) = tempEvaXY(ii)
 do jj=1,kmax
    ListEveXY(pnucl,ii,jj) = tempEveXY(ii,jj)
 end do
end do

do ii = 1,2*NL
   do jj=1,kmax
     ListEveRedXY(pnucl,ii,jj) =
ListEveXY(pnucl,ListLevXY(pnucl,ii),jj)
   end do
end do


!deallocate(tempEva(kmax))
!deallocate(tempEve(kmax,kmax))
!deallocate(tempEveR(kmax,2*NL))
!deallocate(tempS(kmax,kmax))
```

```
!deallocate(WORK(LWORK))

return
end subroutine pair_coeff

!*********************************************************************
**************

subroutine tnucl_coeff
implicit none
integer                 ::l

do l=1,NB
 call nucl_coeff(l)
end do
return
end subroutine tnucl_coeff

!*********************************************************************
***************

subroutine tpair_coeff
implicit none
integer                 ::l

do l=1,NS
 call pair_coeff(l)
end do
return
end subroutine tpair_coeff


!*********************************************************************
**************

subroutine total_data
call input_data
call nucl_data
call tnucl_coeff
call pairs_data
call tpair_coeff
return
end subroutine total_data



subroutine nucl_greenf(mon1,Ei1,r,s,nuclGFrs)
!--------------------------------------------------------------------
-------------------

!--------------------------------------------------------------------
-------------------
```

```
implicit none
character(2),intent(in)                ::mon1
real*8, intent(in)                     ::Ei1
integer,intent(in)                     ::r,s
complex*16,intent(out)                 ::nuclGFrs
integer                                ::indx1,j1
real*8                                 ::numer1
complex*16                             ::denom1
real*8,allocatable,dimension(:,:)      ::eveTrpX

!-------------------------------------------------------------------
------
!Search for thr position of the pair in the ListOrde array for pointer
to the
!multi-dimensional arry that stored the inut data
!-------------------------------------------------------------------
------
do j1=1,NB
  if (ListOrdX(j1)==mon1) then
   indx1=j1
  end if
end do

allocate(eveTrpX(NMAX,NMAX))
eveTrpX = transpose(ListEveX(indx1,:,:))
!-------------------------------------------------------------------
------
!inizialite the variables
!-------------------------------------------------------------------
------
nuclGFrs = cmplx(0.0d00,0.0d00)

!-------------------------------------------------------------------
-----
!the calculation of rs component of the GF
!-------------------------------------------------------------------
-----
do j1=1,ListLHX(indx1)
   numer1 = ListEveX(indx1,r,j1)*eveTrpX(j1,s)
   denom1 = cmplx(Ei1,delta)-ListEvaX(indx1,j1,1)
   nuclGFrs = nuclGFrs + numer1/denom1
end do

!deallocate(eveTrp(NMAX,NMAX))
return
end subroutine

!*******************************************************************
*************************
```

```
subroutine pair_greenf(pair1,Ei2,r,s,pairGFrs)
!-----------------------------------------------------------------------
------------------
!Subroutine that calculate the r,s component of the Green's function
(retarded) of
! a pair of monomers A1A2 defined as <r|G(A1,A2)|s>, where G is the
total GF matrix.
!
!This subroutine take as input the pair of monomers (file), the energy
(Ei), the r,s
!component of the matrix, and return as output the r,s component of
the GF (pairGFrs).
!
!Since the eigenvalues and eigenvectors of the Hamiltonians of each
pair are known, one
!can use the spectral decomposition matrix for calculate the pair GF,
given by the expression:
!
!        <r|G(A1,A2)|s>= sum_k (C(r,k)C*(k,s)/(E-E(k)))
!
!with C(i,j) are the i,j component of the matrix eigenvector,and, E(k)
the eigenvalues
!
!-----------------------------------------------------------------------
--------------------

implicit none
character(2),intent(in)                ::pair1
real*8, intent(in)                     ::Ei2
integer,intent(in)                     ::r,s
complex*16,intent(out)                 ::pairGFrs
integer                                ::indx2,j2
real*8                                 ::numer2
complex*16                             ::denom2
real*8,allocatable,dimension(:,:)      ::eveTrpXY

!-----------------------------------------------------------------------
------
!Search for thr position of the pair in the ListOrdeXY array for
pointer to the
!multi-dimensional arry that stored the inut data
!-----------------------------------------------------------------------
------
do j2=1,NS
  if (ListOrdXY(j2)==pair1) then
   indx2=j2
  end if
end do

allocate(eveTrpXY(NMAX,NMAX))
eveTrpXY = transpose(ListEveXY(indx2,:,:))
```

```
!-----------------------------------------------------------------------
------
!inizialite the variables
!-----------------------------------------------------------------------
------
pairGFrs = cmplx(0.0d00,0.0d00)

!-----------------------------------------------------------------------
-----
!the calculation of rs component of the GF
!-----------------------------------------------------------------------
-----
do j2=1,ListLHXY(indx2)
   numer2 = ListEveXY(indx2,r,j2)*eveTrpXY(j2,s)
   denom2 = cmplx(Ei2,delta)-ListEvaXY(indx2,j2,1)
   pairGFrs = pairGFrs + numer2/denom2
end do

!deallocate(eveTrp(NMAX,NMAX))
return
end subroutine

!**********************************************************************
********************


subroutine rdnucl_greenf(mon2,Ei3,rdnuclGF)
!-----------------------------------------------------------------------
-----------------------
!Subroutine that calculate the reduce Green's function matrix, i.e.
the GF projecte over a
!small set of selected levels. It is a 2x2 block matrix with block
elements, G(Ai,Aj),i,j=1,2,
!where the four block are: G(A1,A1), G(A1,A2), G(A2,A1) and
G(A2,A2).The zise of each block
!NLxNL, with NL is the total number of selected levels.
!
!This subroutine take as input the pair of monomers (file), the energy
(Ei), and return as
! output the  the reduce GF matrix (pairGFrs) at the given energy of
the particular pair.
!
!As in case of the s,r, component of GF, one ca use the spectral
decomposition fo calulate
!the reduce GF of a pais (see the pair_greenf subroitine)
!
!Note: Ai, Aj, with i,j=1,2, means fragment Ai and fragment Aj.
!-----------------------------------------------------------------------
-----------------------

implicit none
character(2),intent(in)                                ::mon2
```

```
real*8,intent(in)                                          ::Ei3
real*8,dimension(NL,NL),intent(out)                        ::rdnuclGF
integer                                                    ::indx3,p1,k1,k2
real*8                                                     ::numer3
real*8                                                     ::denom3
real*8                                                     ::tempgfX
real*8,allocatable,dimension(:,:)                          ::rdeveTrpX

!-------------------------------------------------------------------
------
!Search for thr position of the pair in the ListOrde array for pointer
to the
!multi-dimensional arry that stored the inut data
!-------------------------------------------------------------------
------
do p1=1,NB
  if (ListOrdX(p1)==mon2) then
   indx3=p1
  end if
end do

allocate(rdeveTrpX(NMAX,NL))
rdeveTrpX = transpose(ListEveRedX(indx3,:,:))
!-------------------------------------------------------------------
------
!inizialite the variables
!-------------------------------------------------------------------
------
rdnuclGF = 0.0d00

!-------------------------------------------------------------------
-----
!the calculation of the reduce GF matrix
!-------------------------------------------------------------------
-----
do k1=1,NL
  do k2=1,NL

     do p1=1,ListLHX(indx3)
      numer3  = ListEveRedX(indx3,k1,p1)*rdeveTrpX(p1,k2)
      denom3  = Ei3 - ListEvaX(indx3,p1,1)
     rdnuclGF(k1,k2) = rdnuclGF(k1,k2) + numer3/denom3
     end do

  end do
end do

!deallocate(rdeveTrp(2*NL,NMAX))
return
end subroutine rdnucl_greenf
```

```
!**********************************************************************
************


subroutine rdpair_greenf(pair2,Ei4,rdpairGF)
!----------------------------------------------------------------------
----------------------
!Subroutine that calculate the reduce Green's function matrix, i.e.
the GF projecte over a
!small set of selected levels. It is a 2x2 block matrix with block
elements, G(Ai,Aj),i,j=1,2,
!where the four block are: G(A1,A1), G(A1,A2), G(A2,A1) and
G(A2,A2).The zise of each block
!NLxNL, with NL is the total number of selected levels.
!
!This subroutine take as input the pair of monomers (file), the energy
(Ei), and return as
! output the  the reduce GF matrix (pairGFrs) at the given energy of
the particular pair.
!
!As in case of the s,r, component of GF, one ca use the spectral
decomposition fo calulate
!the reduce GF of a pais (see the pair_greenf subroitine)
!
!Note: Ai, Aj, with i,j=1,2, means fragment Ai and fragment Aj.
!----------------------------------------------------------------------
------------------------

implicit none
character(2),intent(in)                                  ::pair2
real*8,intent(in)                                        ::Ei4
real*8,dimension(2*NL,2*NL),intent(out)                  ::rdpairGF
integer                                                  ::indx4,p1,k1,k2
real*8                                                   ::numer4
real*8                                                   ::denom4
real*8                                                   ::tempgfXY
real*8,allocatable,dimension(:,:)                        ::rdeveTrpXY

!----------------------------------------------------------------------
------
!Search for thr position of the pair in the ListOrde array for pointer
to the
!multi-dimensional arry that stored the inut data
!----------------------------------------------------------------------
------
do p1=1,NS
  if (ListOrdXY(p1)==pair2) then
   indx4=p1
  end if
end do

allocate(rdeveTrpXY(NMAX,2*NL))
```

```
rdeveTrpXY = transpose(ListEveRedXY(indx4,:,:))
!-----------------------------------------------------------------------
------
!inizialite the variables
!-----------------------------------------------------------------------
------
rdpairGF = 0.0d00

!-----------------------------------------------------------------------
-----
!the calculation of the reduce GF matrix
!-----------------------------------------------------------------------
-----
do k1=1,2*NL
  do k2=1,2*NL

     do p1=1,ListLHXY(indx4)
      numer4  = ListEveRedXY(indx4,k1,p1)*rdeveTrpXY(p1,k2)
      denom4  = Ei4 - ListEvaXY(indx4,p1,1)
     rdpairGF(k1,k2) = rdpairGF(k1,k2) + numer4/denom4
     end do

  end do
end do

!deallocate(rdeveTrp(2*NL,NMAX))
return
end subroutine rdpair_greenf

!*********************************************************************
***************


subroutine rdnucl_hamt(mon3,Ei5,rdmonH)
!-----------------------------------------------------------------------
--------------

! ----------------------------------------------------------------------
-----------------

implicit none
character(2),intent(in)                                ::mon3
real*8,intent(in)                                      ::Ei5
real*8,dimension(NL,NL),intent(out)                    ::rdmonH
integer                                                ::indx5,j1
!-----------------------------------------------------------------------
-----
! temporary matrices that stored the isolate hamiltonian and the
reduce GF
!-----------------------------------------------------------------------
-----
!real*8,allocatable,dimension(:,:)                     ::temprdGF0X
```

```
real*8,allocatable,dimension(:,:)                  ::temprdGFX


!-------------------------------------------------------------------
-----
!input for inversion with lapack[1] library (zgetrf and zgetri
subroutines)
!-------------------------------------------------------------------
-----
real*8,allocatable,dimension(:)                     ::WORK3
integer,allocatable,dimension(:)                    ::ipvt3
integer                                             ::info3,LWORK3
!real*8,dimension(NL,NL)                             ::idt3
external dgetrf
external dgetri

LWORK3=max(1,NL)

!-------------------------------------------------------------------
------
!Allocate the memory
!-------------------------------------------------------------------
------
!allocate(temprdGF0X(NL,NL))
allocate(temprdGFX(NL,NL))
allocate(WORK3(LWORK3))
allocate(ipvt3(NL))

!-------------------------------------------------------------------
------
!inizialite the variables
!-------------------------------------------------------------------
------
rdmonH = 0.0d00
!temprdGF0X = 0.0d00
!idt3 = 0.0d00

!-------------------------------------------------------------------
------
!call the rdpair_greenf subroutine to get the rdpairGF
!-------------------------------------------------------------------
------
call rdnucl_greenf(mon3,Ei5,temprdGFX)

!-------------------------------------------------------------------
------
!Search for thr position of the pair in the ListOrde array for pointer
to the
!multi-dimensional arry that stored the inut data
!-------------------------------------------------------------------
------
do j1=1,NB
```

```
  if (ListOrdX(j1)==mon3) then
   indx5=j1
  end if
end do

!do j1=1,NL
!idt3(j1,j1)=1.00d00
!end do


!-------------------------------------------------------------------------
-----
!the calculation of g(Ai,Aj)^-1 = E-H0(Ai,Aj)
!-------------------------------------------------------------------------
-----

!  temprdGF0X = Ei5*ListSRedX(indx5,:,:)


!-------------------------------------------------------------------------
------
!inversion of reduce pair GF
!-------------------------------------------------------------------------
------
call dgetrf(NL, NL,temprdGFX,NL, ipvt3, info3)
if (info3 .gt.0) then
  write(*,*)'ERROR:,failed to compute IP list for inversion of
rdGFX',mon3
 stop
end if

call dgetri(NL,temprdGFX,NL,ipvt3,WORK3, LWORK3, info3)
if (info3 .gt.0) then
  write(*,*)'ERROR:,failed to compute the inverse matrix of
rdGFX',mon3
 stop
end if

!-------------------------------------------------------------------------
-----
!calculation of reduce perturbation matrix rdpairV
!-------------------------------------------------------------------------
-----
rdmonH =  Ei5*ListSRedX(indx5,:,:) - temprdGFX

!-------------------------------------------------------------------------
-----
!dealocate the memory
!-------------------------------------------------------------------------
-----
!deallocate(temprdGF0(2*NL,2*NL))
!deallocate(temprdGF(2*NL,2*NL))
```

```
!deallocate(WORK1(LWORK1))
!deallocate(ipvt1(2*NL))

return

end subroutine rdnucl_hamt

!*********************************************************************
****************************

subroutine total_rdnuclH
implicit none
integer                                  ::q

allocate(rdVNSX(NB,NL,NL))
do q=1, NB
 call rdnucl_hamt(ListOrdX(q),test_E,rdVNSX(q,:,:))
end do
return
end subroutine total_rdnuclH


subroutine rdpair_pert(pair3,Ei6,rdpairV)
!-------------------------------------------------------------------
---------------
!Subroutine compute the reduce pair perturbation matrix come from the
decouplin
!the set  of selected levels of the total Fock matrix Hamiltonian.
!
!The subroutine take as input variables the pari of monomers
(file),the energy(Ei) .
!and return as output the reduce perturbation matrix (rdpairV)
!
!The rdpairV is a 2x2 block matrix, with each block of zise NLxNL,
given by

!               V(Ai,Aj)=g(Ai,Aj)^-1 -G(Ai,Aj)^-1
!
!where NL is the number of levels, g(Ai,Aj) the isolate reduce GF, and
G(Ai,Aj) the
!redduce GF matirx of pair AiAj.

!Note: Ai, Aj, with i,j=1,2, means fragment Ai and fragment Aj.
! ------------------------------------------------------------------
-----------------

implicit none
character(2),intent(in)                                  ::pair3
real*8,intent(in)                                        ::Ei6
real*8,dimension(2*NL,2*NL),intent(out)                  ::rdpairV
integer                                                  ::indx6,indx7,j1
```

```
!-----------------------------------------------------------------------
-----
! temporary matrices that stored the isolate hamiltonian and the
reduce GF
!-----------------------------------------------------------------------
-----
real*8,allocatable,dimension(:,:)                  ::temprdVXY
real*8,allocatable,dimension(:,:)                  ::temprdGFXY


!-----------------------------------------------------------------------
-----
!input for inversion with lapack[1] library (zgetrf and zgetri
subroutines)
!-----------------------------------------------------------------------
-----
real*8,allocatable,dimension(:)                     ::WORK4
integer,allocatable,dimension(:)                    ::ipvt4
integer                                             ::info4,LWORK4
!real*8,dimension(2*NL,2*NL)                         ::idt4
external dgetrf
external dgetri

LWORK4=max(1,2*NL)

!-----------------------------------------------------------------------
------
!Allocate the memory
!-----------------------------------------------------------------------
------
allocate(temprdVXY(2*NL,2*NL))
allocate(temprdGFXY(2*NL,2*NL))
allocate(WORK4(LWORK4))
allocate(ipvt4(2*NL))

!-----------------------------------------------------------------------
------
!inizialite the variables
!-----------------------------------------------------------------------
------
rdpairV = 0.0d00
temprdVXY = 0.0d00
!idt4 = 0.0d00

!-----------------------------------------------------------------------
------
!call the rdpair_greenf subroutine to get the rdpairGF
!-----------------------------------------------------------------------
------
call rdpair_greenf(pair3,Ei6,temprdGFXY)
```

```
!-----------------------------------------------------------------------
------
!Search for thr position of the pair in the ListOrde array for pointer
to the
!multi-dimensional arry that stored the inut data
!-----------------------------------------------------------------------
------
do j1=1,NS
  if (ListOrdXY(j1)==pair3) then
   indx6=j1
  end if
end do

!do j1=1,2*NL
!idt4(j1,j1)=1.00d00
!end do


 do j1=1,NB
   if(pair3(1:1)==ListOrdX(j1))then
     indx7=j1
   end if
end do

temprdVXY(1:NL,1:NL) = rdVNSX(indx7,:,:)

do j1=1,NB
   if(pair3(2:2)==ListOrdX(j1))then
     indx7=j1
   end if
end do

temprdVXY(NL+1:2*NL,NL+1:2*NL) = rdVNSX(indx7,:,:)


!-----------------------------------------------------------------------
-----
!the calculation of g(Ai,Aj)^-1 = E-H0(Ai,Aj)
!-----------------------------------------------------------------------
-----

  temprdVXY = Ei6*ListSRedXY(indx6,:,:) - temprdVXY


!-----------------------------------------------------------------------
------
!inversion of reduce pair GF
!-----------------------------------------------------------------------
------
call dgetrf(2*NL, 2*NL,temprdGFXY,2*NL, ipvt4, info4)
if (info4 .gt.0) then
```

```
  write(*,*)'ERROR:,failed to compute IP list for inversion of
rdGFXY',pair3
 stop
end if

call dgetri(2*NL,temprdGFXY,2*NL,ipvt4,WORK4, LWORK4, info4)
if (info4 .gt.0) then
  write(*,*)'ERROR:,failed to compute the inverse matrix of
rdGFXY',pair3
 stop
end if

!---------------------------------------------------------------------
-----
!calculation of reduce perturbation matrix rdpairV
!---------------------------------------------------------------------
-----
rdpairV = temprdVXY - temprdGFXY

!---------------------------------------------------------------------
-----
!dealocate the memory
!---------------------------------------------------------------------
-----
!deallocate(temprdGF0(2*NL,2*NL))
!deallocate(temprdGF(2*NL,2*NL))
!deallocate(WORK1(LWORK1))
!deallocate(ipvt1(2*NL))

return

end subroutine rdpair_pert

!*********************************************************************
******************

subroutine total_rdpert
implicit none
integer                                 ::q

call total_rdnuclH

allocate(rdVNSXY(NS,2*NL,2*NL))
do q=1, NS
 call rdpair_pert(ListOrdXY(q),test_E,rdVNSXY(q,:,:))
end do
return
end subroutine total_rdpert

!*********************************************************************
******************
```

```
!*********************************************************************
***************


subroutine total_hamt
!---------------------------------------------------------------------
------
!Purpose:
!Subroutine for calculated the total Hamiltonian of a chain of
molecule of
!length N into the TB model with first NN interaction.
!one start we the sequencen of molecules : A1-A2-A3--------Ai--------
AN-1-AN;

!the total Hamiltonian and Overlap matrix of the chain can write as a
NxN block
!matrix with each block define as Hij=H(Ai,Aj),  means the block matix
belong
!to the molecule in the site i and the molecule in the site j.
!
!Into the first NN interaction TB model these blocks can be write as
!
!    Hii = H0(Ai)+ V11(Ai-1,Ai)+ V11(Ai,Ai+1)      i=1,N
!    Hi,i+1 = V12(Ai,Ai+1)+ V21(Ai,Ai+1)           i=1,N-1
!    Hi+1,i = V12(Ai+1,Ai)+ V21(Ai+1,Ai)           i=1,N-1
!    Hij = 0 fof i .ne.j and abs(i-j) .ne.1        i=1,N and j=1,N
!
!    Sii = 1                                       i=1,N
!    Si,i+1 = S12(Ai,Ai+1)+ S21(Ai,Ai+1)           i=1,N-1
!    Si+1,i = S12(Ai+1,Ai)+ S21(Ai+1,Ai)           i=1,N-1
!    Sij = 0 fof i .ne.j and abs(i-j) .ne.1        i=1,N and j=1,N
!
!
!Where H0(Ai) is the isolate hamiltonian of the molecule in site i,
V11(Ai-1,Ai), V11(Ai,Ai+1)
!V12(Ai,Ai+1) and V21(Ai,Ai+1)are the block component of the
perturbation hamiltonian of pair
!wiht molecules in sites i-1 and i, and pair with molecule in sites i
and i+1.

!Note: remember that all these block are martice of zise NlxNl, with
nl the number of
!levels taken for the reduce representation
!
! Record of revisions
!   Date      Programer          Descripion of change
! -------   --------------   ------------------------------------
! 02/30/2023   J. R. Alvarez    Original code
implicit none
real*8,allocatable,dimension(:,:)             ::tH0
integer                                       ::indx8,indx9
```

```
integer                                          ::i,j

allocate(totalH(NL*NM,NL*NM))
allocate(tH0(NL*NM,NL*NM))

call total_rdpert

tH0 = 0.0d00
totalH = 0.0d00

do i=1,NM
do j=1,NB
  if (Seq(i) == ListOrdX(j)) then
   indx8=j
  end if
end do

 tH0((i-1)*NL+1:i*NL,(i-1)*NL+1:i*NL)= rdVNSX(indx8,:,:)

end do



do j=1,NS
  if (Coupl(1) == ListOrdXY(j)) then
   indx9=j

totalH(1:2*NL,1:2*NL)=rdVNSXY(indx9,1:2*NL,1:2*NL)

 else if
(Coupl(1)==ListOrdXY(j)(2:2)//ListOrdXY(j)(1:1).and.ListOrdXY(j)(1:1)/
=ListOrdXY(j)(2:2)) then
  indx9=j

totalH(1:NL,1:NL)=rdVNSXY(indx9,NL+1:2*NL,NL+1:2*NL)
totalH(NL+1:2*NL,NL+1:2*NL)=rdVNSXY(indx9,1:NL,1:NL)
totalH(1:NL,NL+1:2*NL)=rdVNSXY(indx9,1:NL,NL+1:2*NL)
totalH(NL+1:2*NL,1:NL)=rdVNSXY(indx9,NL+1:2*NL,1:NL)

  end if
end do


do i=2,NM-1
 do j=1,NS
  if(Coupl(i)==ListOrdXY(j)) then
  indx9=j

  totalH((i-1)*NL+1:i*NL,(i-1)*NL+1:i*NL) = &
  totalH((i-1)*NL+1:i*NL,(i-1)*NL+1:i*NL) + rdVNSXY(indx9,1:NL,1:NL)
```

```
  totalH((i-1)*NL+1:i*NL,i*NL+1:(i+1)*NL) =
rdVNSXY(indx9,1:NL,NL+1:2*NL)
  totalH(i*NL+1:(i+1)*NL,(i-1)*NL+1:i*NL) =
rdVNSXY(indx9,NL+1:2*NL,1:NL)

  totalH(i*NL+1:(i+1)*NL,i*NL+1:(i+1)*NL) =
rdVNSXY(indx9,NL+1:2*NL,NL+1:2*NL)


  else if
(Coupl(i)==ListOrdXY(j)(2:2)//ListOrdXY(j)(1:1).and.ListOrdXY(j)(1:1)/
=ListOrdXY(j)(2:2)) then
  indx9=j

  totalH((i-1)*NL+1:i*NL,(i-1)*NL+1:i*NL) = &
  totalH((i-1)*NL+1:i*NL,(i-1)*NL+1:i*NL) +
rdVNSXY(indx9,NL+1:2*NL,NL+1:2*NL)


  totalH((i-1)*NL+1:i*NL,i*NL+1:(i+1)*NL) =
rdVNSXY(indx9,1:NL,NL+1:2*NL)
  totalH(i*NL+1:(i+1)*NL,(i-1)*NL+1:i*NL) =
rdVNSXY(indx9,NL+1:2*NL,1:NL)

  totalH(i*NL+1:(i+1)*NL,i*NL+1:(i+1)*NL) =   rdVNSXY(indx9,1:NL,1:NL)

  end if
 end do
end do

do j=1,NS
  if (Coupl(NM)==ListOrdXY(j)) then
   indx9=j

  totalH((NM-1)*NL+1:NM*NL,(NM-1)*NL+1:NM*NL) = &
  totalH((NM-1)*NL+1:NM*NL,(NM-1)*NL+1:NM*NL) +
rdVNSXY(indx9,1:NL,1:NL)

  totalH(1:NL,(NM-1)*NL+1:NM*NL) = rdVNSXY(indx9,NL+1:2*NL,1:NL)

  totalH((NM-1)*NL+1:NM*NL, 1:NL) = rdVNSXY(indx9,1:NL,NL+1:2*NL)

  totalH(1:NL,1:NL) = &
  totalH(1:NL,1:NL) + rdVNSXY(indx9,NL+1:2*NL,NL+1:2*NL)

 else if
(Coupl(NM)==ListOrdXY(j)(2:2)//ListOrdXY(j)(1:1).and.ListOrdXY(j)(1:1)
/=ListOrdXY(j)(2:2)) then
  indx9=j

  totalH((NM-1)*NL+1:NM*NL,(NM-1)*NL+1:NM*NL) = &
```

```
  totalH((NM-1)*NL+1:NM*NL,(NM-1)*NL+1:NM*NL) +
rdVNSXY(indx9,NL+1:2*NL,NL+1:2*NL)

  totalH(1:NL,(NM-1)*NL+1:NM*NL) = rdVNSXY(indx9,NL+1:2*NL,1:NL)

  totalH((NM-1)*NL+1:NM*NL, 1:NL) = rdVNSXY(indx9,1:NL,NL+1:2*NL)

  totalH(1:NL,1:NL) = &
  totalH(1:NL,1:NL) + rdVNSXY(indx9,1:NL,1:NL)


  end if
end do


totalH= totalH + tH0

return
end subroutine total_hamt

!*********************************************************************
********

subroutine total_overlp

implicit none
!real*8,allocatable,dimension(:,:)          ::totalS
integer                                     ::indx10
integer                                     ::i,j

allocate(totalS(NL*NM,NL*NM))

totalS=0.0d00

do i=1,NM-1
do j=1,NS
  if (Coupl(i) == ListOrdXY(j)) then
   indx10=j
    totalS((i-1)*NL+1:i*Nl,i*NL+1:(i+1)*Nl)=
ListSRedXY(indx10,1:NL,NL+1:2*NL)
    totalS(i*NL+1:(i+1)*Nl,(i-1)*NL+1:i*Nl)=
ListSRedXY(indx10,NL+1:2*NL,1:NL)

 else if
(Coupl(i)==ListOrdXY(j)(2:2)//ListOrdXY(j)(1:1).and.ListOrdXY(j)(1:1)/
=ListOrdXY(j)(2:2)) then
  indx10=j

 totalS((i-1)*NL+1:i*Nl,i*NL+1:(i+1)*Nl)=
ListSRedXY(indx10,1:NL,NL+1:2*NL)
 totalS(i*NL+1:(i+1)*Nl,(i-1)*NL+1:i*Nl)=
ListSRedXY(indx10,NL+1:2*NL,1:NL)
```

```
  end if
end do
end do

do j=1,NS
if (Coupl(NM) == ListOrdXY(j)) then
   indx10=j
    totalS(1:NL,(NM-1)*NL+1:NM*NL)= ListSRedXY(indx10,1:NL,NL+1:2*NL)
    totalS((NM-1)*NL+1:NM*NL,1:NL)= ListSRedXY(indx10,NL+1:2*NL,1:NL)

  else
if(Coupl(NM)==ListOrdXY(j)(2:2)//ListOrdXY(j)(1:1).and.ListOrdXY(j)(1:
1)/=ListOrdXY(j)(2:2)) then
    indx10=j
    totalS(1:NL,(NM-1)*NL+1:NM*NL)= ListSRedXY(indx10,NL+1:2*NL,1:NL)
    totalS((NM-1)*NL+1:NM*NL,1:NL)= ListSRedXY(indx10,1:NL,NL+1:2*NL)
end if
end do

do i=1,NM*NL
totalS(i,i)=1.00d00
end do

return
end subroutine total_overlp

!*********************************************************************
**********

!*********************************************************************
**********

!*********************************************************************
**********

!*********************************************************************
**********

end module INIDATA
```

```
module TRANSPORT
use INIDATA


!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

contains

subroutine total_greenf0(Ej3,totalGF0)
implicit none
real*8,intent(in)                                          ::Ej3
complex*16,dimension(NM*NL,NM*NL),intent(out)              ::totalGF0
integer                                                    ::i,j
real*8,dimension(NM*NL,NM*NL)                              ::ident1
complex*16,allocatable,dimension(:)                        ::WORK5
integer,allocatable,dimension(:)                           ::ipvt5
integer::stat
integer::info5,LWORK5


LWORK5 = max(1,NM*NL)

allocate(WORK5(LWORK5))
allocate(ipvt5(NM*NL))

ident1=0.0d00

do i=1,NM*NL
    ident1(i,i)=1.0d00
end do

totalGF0 = cmplx(Ej3,delta)*totalS - totalH

call zgetrf(NM*NL,NM*NL,totalGF0,NM*NL, ipvt5, info5)
if (info5 .gt.0) then
  print *,'ERROR:,failed to compute the PI values for totalGF0'
 stop
end if
call zgetri(NM*NL,totalGF0,NM*NL,ipvt5,WORK5, LWORK5, info5)
if (info5 .gt.0) then
  print *,'ERROR:,failed tocompute the inverse of totalGF0'
 stop
end if

!deallocate(temp_tH(NM*NL,NM*NL))
!deallocate(temp_tS(NM*NL,NM*NL))
```

```
!deallocate(WORK2)
!deallocate(ipvt2)

return
end subroutine total_greenf0

!**********************************************************************
********************

subroutine couple_greenf(Ek1,SiteGF)
implicit none
real*8 ,intent(in)                              ::Ek1
complex*16,dimension(NM,NL,NL),intent(out)  ::SiteGF
integer                                         ::site,i,j
complex*16, allocatable, dimension (:,:)    ::temptGF0,tempSGF
real*8, allocatable, dimension (:,:)        ::ident

complex*16,allocatable,dimension(:)         ::WORK6
integer,allocatable,dimension(:)            ::ipvt6
integer                                         ::stat,status
integer                                         ::info6,LWORK6



LWORK6=max(1,NL)

allocate(temptGF0(NM*NL,NM*NL))
allocate(ident(NL,NL))
allocate(tempSGF(NL,NL))
allocate(WORK6(LWORK6))
allocate(ipvt6(NL))

do i=1,NL
  do j=1,NL
  if (i==j)then
    ident(i,i)=1.0d00
  else
    ident(i,j)=0.0d00
   end if
  end do
end do


call total_greenf0(Ek1,temptGF0)

do site=1,NM

 tempSGF = temptGF0((site-1)*NL+1:site*NL,(site-1)*NL+1:site*NL)
 tempSGF = ident + cmplx(0.0d00,0.5d00*Gm)*tempSGF

call zgetrf(NL, NL,tempSGF,NL, ipvt6, info6)
 if (info6 .gt.0) then
```

```
   print *,'ERROR:,failed to compute IP values for site GF'
  stop
 end if

call zgetri(NL,tempSGF,NL,ipvt6,WORK6, LWORK6, info6)
 if (info6 .gt.0) then
   print *,'ERROR:,failed to compute the inverse for site GF'
  stop
 end if

 tempSGF = matmul(tempSGF,&
        temptGF0((site-1)*NL+1:site*NL,(site-1)*NL+1:site*NL))

 SiteGF(site,:,:) = tempSGF

end do

!deallocate(temptGF0(NM*NL,NM*NL))
!deallocate(tempSGF(NL,NL))
!deallocate(ident(NL,NL))
!deallocate(WORK3(NL))
!deallocate(ipvt3(NL))

return
end subroutine couple_greenf

!**********************************************************************
**************

subroutine fermi_dirac(x,FermiL,FermiR)
!----------------------------------------------------------------------
-
!Fermi-Dirac subroutine, for a given energy (input),it calculate
!the occupation in the left (fermiL) and rigth (fermiR) electrode.
!subroutine take as input the energy (x), and return as output
!the value of the Fermi function in the left and right electrode
!(fermiL,fermiR)
!----------------------------------------------------------------------
-
implicit none
real*8                   ::x
real*8                   ::FermiL
real*8                   ::FermiR

FermiL=1.0d0/((exp(x-MuL)/(Kb*Tp))+1.0d0)
FermiR=1.0d0/((exp(x-MuR)/(Kb*Tp))+1.0d0)

return
end subroutine fermi_dirac

!**********************************************************************
***************
```

```
subroutine transm_func(Ek2,SiteTransm)
implicit none
!integer                                    ::site
real*8,intent(in)                           ::Ek2
real*8,dimension(NM),intent(out)            ::SiteTransm
real*8,allocatable,dimension(:)             ::transmF
complex*16,allocatable,dimension(:,:,:)     ::tempSiteGF
complex*16,allocatable,dimension(:,:)       ::tempst
integer                                     ::j,k



allocate(tempSiteGF(NM,NL,NL))
allocate(transmF(NM))
allocate(tempst(NL,NL))


call couple_greenf(Ek2,tempSiteGF)

do k=1,NM
  tempst = tempSiteGF(k,:,:)
 transmF(k) = 0.0d00
 tempst = matmul(tempst,conjg(transpose(tempst)))

do j=1,NL
    transmF(k)=transmF(k)+ tempst(j,j)
  end do
end do

 SiteTransm = ((Gm/2.0d00)**2)*transmF

!deallocate(tempSiteGF(NM,NL,NL))
!deallocate(transmF(NM))
!deallocate(tempst(NL,NL))

return
end subroutine transm_func

!*****************************************************************
*****************



subroutine curr_intg(Ek3,intg)
implicit none
real*8,intent(in)                   ::Ek3
real*8,dimension(NM),intent(out)    ::intg
real*8,allocatable,dimension(:)     ::tempTr
real*8                              ::fleft,fright
integer                             ::j

allocate(tempTr(NM))
```

```
call fermi_dirac(Ek3,fleft,fright)
call transm_func(Ek3,tempTr)

do j=1,NM
  intg(j)=(fleft-fright)*tempTr(j)
end do

!deallocate(tempTr(NM))
return
end subroutine curr_intg

!*********************************************************************
***************


integer function integrand(ndim, xx, ncomp, ff)
implicit none
integer                   ::ndim,ncomp
real*8                    :: xx(ndim), ff(ncomp)
real*8                    ::lower,upper


call curr_intg((MuR-MuL)*xx(1)+MuL,ff)

ff=(MuR-MuL)*ff

!integrand=0
return
end function integrand

!*********************************************************************
****

subroutine land_curr(LandCurr)
implicit none
real*8,dimension(NM),intent(out)   ::LandCurr
integer                            ::j
!-------------------------------------------------------------
!parameters for cuba library
!-------------------------------------------------------------

integer,parameter                  :: ndim = 2
integer                            :: ncomp
real*8,parameter                   :: userdata = 0
integer,parameter                  :: nvec = 1
real*8,parameter                   :: epsrel = 1D-3
real*8,parameter                   :: epsabs = 1D-10
integer,parameter                  :: seed = 0
integer,parameter                  :: mineval = 0
integer,parameter                  :: maxeval = 50000
integer,parameter                  :: nstart = 1000
```

```
integer,parameter                   :: nincrease = 500
integer,parameter                   :: nbatch = 1000
integer,parameter                   :: gridno = 0
character*(*),parameter             :: statefile = ""
real*8,dimension(NM)                :: integral, error, prob
integer                             :: verbose, nregions, neval, fail
character*16                        ::env
integer*8,parameter                 ::spin = -1
integer,parameter                   ::last = 4
real*8,parameter                    ::flatness = 25.0d00
integer,parameter                   ::nnew = 1000
integer,parameter                   ::key = 0

ncomp = NM



!call getenv("CUBAVERBOSE",env)
!verbose=1
!read(env,*,iostat=fail,end=999,err=999)verbose
!999 continue

!select case(IntSub)
!case(1)
!call vegas(ndim,ncomp,integrand,userdata,nvec,&
!      epsrel,epsabs,verbose,seed,&
!      mineval,maxeval,nstart,nincrease,nbatch,&
!      gridno,statefile,spin,&
!      neval,fail,integral,error,prob)

!case(2)
!call suave(ndim, ncomp, integrand, userdata, nvec,&
!         epsrel, epsabs, verbose + last, seed,&
!         mineval, maxeval, nnew, flatness,&
!         statefile, spin,&
!         nregions, neval, fail, integral, error, prob)

!case(3)
!call divone(ndim, ncomp, integrand, userdata, nvec,&
!         epsrel, epsabs, verbose + last, seed,&
!         mineval, maxeval, nnew, flatness,&
!         statefile, spin,&
!         nregions, neval, fail, integral, error, prob)

!case default
call cuhre(ndim, ncomp, integrand, userdata, nvec,&
         epsrel, epsabs, verbose + last,&
         mineval, maxeval, key,&
         statefile, spin,&
        nregions, neval, fail, integral, error, prob)

!end select
```

```
do j=1,NM
LandCurr(j) = integral(j)
end do

return
end subroutine land_curr

!**********************************************************************
***************

!**********************************************************************
*********************
end module TRANSPORT
```