UNIVERSITAT POLITÈCNICA DE CATALUNYA

Departament d'Enginyeria Electrònica

SIMULACIÓN MONTE CARLO DE TRANSISTORES BIPOLARES DE HETEROUNIÓN ABRUPTA (HBT)

Autor: Pau Garcias Salvà
Director: Lluís Prat Viñas

Apéndice 1. Código fuente del simulador MCHBT.

```
ceeee+i**#**** ****#**** ****#**** ****#**** ****#**** ****#**** ****#72------80
c
c*************************  program MCHBT  **************************
c--------------------------------------------------------------------
c
c          FORTRAN program to simulate one dimensional (1D)
c          Heterojunction Bipolar Transistors (HBTs) using
c                    Monte Carlo methods (MC).
c
c                 by Pau Garcias i Salva'. July, 1997
c
c--------------------------------------------------------------------
      program MCHBT
c
c     storage allocation
c
c         implicit double precision (a-h,o-z)
c         implicit integer (i-n)

      implicit none

         integer
     +      ikx,iky,ikz,its,ix,ival,icel,iwgt,ixm,idx,ixup
         parameter (ikx=1,iky=2,ikz=3,its=4,ix=5)
         parameter (ival=1,icel=2,iwgt=3,ixm=1,idx=18,ixup=19)
c
         integer
     +      nbmax,nmax,ntmax,ieTmax,
     +      nptmax,Lrnc,LNorm,ivmax,iregnmax,ismmax,ienmax,
     +      base,base1,inmax,iwmax,npcmin,npcmax,lfrmax,
     +      Lrns,Lbck,L2i,nthdsmax,LB,nptcon,nptcon4

c     L2i = number of integers (4 B) to fill a secondary cache line (128 B)
c     nthdsmax = max. number of threads to be used when parallelizing
c     LB = chunk size for cyclic distribution of data & iterations between
c           processors
c     Condition to avoid false sharing:
c       pt(5,n)  --> LB * 8 (B/elt.) * 5 (elts.) = k1 * 128 (B/cache line L2)
c       ipt(3,n) --> LB * 4 (B/elt.) * 3 (elts.) = k2 * 128 (B/cache line L2)
c     hence: [ k1= 10/3 k2 ] .AND. [ LB = 16/5 k1 ]   with k1, k2 = integers
c     For instance:  LB=32    (taking k1=10 and k2= 3)
c     For instance:  LB=128   (taking k1=40 and k2=12)

         parameter (L2i=32,nthdsmax=8,LB=128)

         parameter (nbmax=3,nmax=430,ntmax=3*nmax+4)
         parameter (npcmin=500,npcmax=600,lfrmax=nmax)
         parameter (base=10,base1=base-1,inmax=2*npcmax)
         parameter (iwmax=9)
         parameter (ieTmax=4000)
         parameter (nptmax=1.1*npcmax*nmax,Lrnc=5,LNorm=40)
         parameter (ivmax=2,iregnmax=3,ismmax=10,ienmax=2000)
         parameter (nptcon=2*npcmax,nptcon4=4*nptcon)

c     Lbck=max. Block Length of r.n.'s generated at every subr. call
c     (Lrns=4, number of r.n.'s needed at
c      every scattering event and re-setting of the next scatt. time)
c     Condition to avoid false sharing:  n * L2sp = 1+3+Lrns*Lbck+24
c       where L2sp = capacity of one 2nd-level cache line= 32 REAL*4
c             1+24 = carry+24 seeds of the r.n. generator
c                3    = dummy value to make possible the equality
         parameter (Lrns=4,Lbck=105)

c     Condition to avoid false sharing in rnc():  n * L2sp = 1+2+Lrnc+24

c        parameter (Lrwk=5,Liwk=2)

         DOUBLE PRECISION MP
         PARAMETER (MP = 1.0D-300)

         double precision pi,q,hbar,bk,eps0,efm0
         double precision Cnorm,Dnorm,Vtnorm,tnorm,xnorm,
     +      vnorm,Enorm,Knorm,Fnorm,Alfnorm,jnorm,
     +      He,Hk,Hv,Hke,deT
         parameter (pi   = 3.14159265d+00, q    = 1.6021892d-19)
         parameter (hbar = 0.65852d-15,     bk   = 8.61738d-05)
         parameter (eps0 = 8.85419d-14,     efm0 = 9.10953d-31)
c
         parameter (Cnorm = 1.0d+20)
         parameter (Dnorm = 1.0d+00)
         parameter (Vtnorm = 300.0*bk)
         parameter (tnorm = (eps0*Vtnorm/q/Cnorm)/Dnorm)
         parameter (xnorm = sqrt(tnorm*Dnorm))
         parameter (vnorm = xnorm/tnorm)
         parameter (Enorm = Vtnorm)
         parameter (knorm = 1.0/xnorm)
         parameter (Fnorm = Vtnorm/xnorm)
         parameter (Alfnorm = 1.0/Enorm)
         parameter (jnorm = (q*Cnorm)/xnorm)

         parameter (He=0.5*hbar*hbar/efm0/Enorm*knorm*knorm*q*1.d+4)
         parameter (Hk = Vtnorm*tnorm/hbar)
         parameter (Hv = q*hbar*1.0d+04/Dnorm/efm0)
         parameter (Hke = 1.0/sqrt(He))

         parameter (deT = 1.0d-3/Enorm)
c
c     work arrays (integer and real)
c
c         integer
c    +       iwka(Liwk*nptmax)
c         double precision
c    +      rwka(Lrwk*nptmax)

        integer
     +      tiphbt,matc,matb,mate,M,ipi(2),ibt,j1,jh(ivmax)
        INTEGER
     +      IMI(2),ITERM,ITER,TIP
        double precision
     +      fmc,fmb,fme,tempK,t(3),psi0(2),efer,par(19,0:nmax+1),
     +      gradp(0:nmax+1),w(nbmax,nmax),wi(1,nmax),nie(2,nmax),
     +                   Sw(nbmax,nmax),
     +      cn0c,cn0e,wght(0:iwmax),rdx,
     +      cn(0:nmax+1),cnaux(0:nmax+1,nthdsmax),Scn(nmax),
     +      cn2(0:nmax+1),cn2aux(0:nmax+1,nthdsmax),Scn2(nmax),
     +      vxm(0:nmax+1),vxmaux(0:nmax+1,nthdsmax),Svx(nmax),
     +      ekm(ivmax,0:nmax+1),ekmaux(ivmax,0:nmax+1,nthdsmax),
     +                   Sek(ivmax,nmax),
     +      fd1(0:nmax+1),fd1aux(0:nmax+1,nthdsmax),Sfd1(nmax),
     +      pdt0(nmax),
     +      Teb(ieTmax,ivmax),E0(ivmax),
     +          mnhbt(nmax),mphbt(nmax),
     +          Ecrie(3),vsate(3),Ecrih(3),vsath(3),
     +          IR(nmax),IRCOL,IRTOTAL
          DOUBLE PRECISION
     +          PARINT(5,2),FET(5,4),refp(5),
     +          ptt,pttc,ni,s0,as,aco,aba,area,rc,rb,re,
     +          xcele(nmax),xchue(nmax),jcol,jelec,tic,vbe,
     +          S(ntmax),B(nmax),C(nmax),
     +          LPROX,ERROR,CORRE,CORL,FNC,FNV
          integer
     +          iseedc,iseeds,jTh,jTm1,jTl1,jTm2,jTl2,jTm3,jTl3,iemax,
     +          La(L2i,nthdsmax),nthds,jdots,frepe,jfrequen
          double precision
     +          parNorm(LNorm),time,dt,de,del,epp,xlow,xhigh,factorj
          integer
     +          npt,npti(0:nmax+1),ipt(3,nptmax),
     +          nptiw(0:iwmax,0:nmax+1),nptnw(inmax,0:iwmax,0:nmax+1),
     +          nptiwaux(0:iwmax,0:nmax+1,nthdsmax),
     +          nptnwaux(inmax,0:iwmax,0:nmax+1,nthdsmax),
     +          ifree(base1,lfrmax),
     +          jelim(nptcon4),joutc(nptcon),joute(nptcon),
     +          joutcaux(0:nptcon,nthdsmax),
     +          jouteaux(0:nptcon,nthdsmax),
     +          nqj(nmax,ivmax),nqjaux(nmax,ivmax,nthdsmax),
     +          nqjt(4,ivmax),nqjtaux(L2i,ivmax,nthdsmax)

          double precision
     +          pt(5,nptmax),
     +          efm(ivmax,iregnmax),alf(ivmax,iregnmax),
     +          gm(ivmax,iregnmax),
     +          swk(ivmax,ismmax,ienmax,iregnmax),
     +          ec(ivmax,iregnmax),
     +          hwo(iregnmax),hwij(iregnmax),hwe(iregnmax)

          real*4
     +          rnc(0:2+Lrnc+24,nthdsmax),
     +          rns(0:3+Lrns*Lbck+24,nthdsmax)
          integer
     +          NFDD,NFinp,NFpMC,NFout,NFout2,NFout3,
     +          NFout4,NFout5,NFout6,NFout7,NFout8,NFout9,NFout0,
     +          NFout7a,NFout2a,NFoutjt

          logical
     +          zexist,znqj
          character*16
     +          fileout

          double precision
     +          FERM12

          integer
     +          lfrtop,npttop,nptitop

          integer
     +          i,j,jT,mmm,lll,iv,n


          external FERM12
c$        integer mp_numthreads

c  data distribution


c$distribute_reshape pt(*,cyclic(LB))
c$distribute_reshape ipt(*,cyclic(LB))
c$distribute_reshape rns(*,cyclic(1))
c$distribute_reshape rnc(*,cyclic(1))
c$distribute_reshape cnaux(*,cyclic(1))
c$distribute_reshape cn2aux(*,cyclic(1))
c$distribute_reshape vxmaux(*,cyclic(1))
c$distribute_reshape fdlaux(*,cyclic(1))
c$distribute_reshape ekmaux(*,*,cyclic(1))
c$distribute_reshape nptiwaux(*,*,cyclic(1))
c$distribute_reshape nptnwaux(*,*,*,cyclic(1))
c$distribute_reshape joutcaux(*,cyclic(1))
c$distribute_reshape jouteaux(*,cyclic(1))
c$distribute_reshape La(*,cyclic(1))
c!!!$distribute_reshape nqjaux(*,*,cyclic(1))
c!!!$distribute_reshape nqjtaux(*,*,cyclic(1))

c!!!! per tal d'evitar comparticio falsa de dades, cal assegurar que
c!!!!  nmax > M + (L2i-1)       (matrius enteres cnaux, vxmaux, ...)
c!!!! o alguna relacio similar, segons cada cas
c!!!! En cas contrari, cal redefinir les dimensions de les matrius, tal
c!!!!  com s'ha fet amb pt, ipt, rns, rnc, ...

c  find out the number of threads assigned to perform the present execution

          nthds = 1
c$        nthds = mp_numthreads()
c$        if (nthds .gt. nthdsmax) STOP 'Too many threads, babe!'

c  define/assign unit number to files

          NFDD=20
          NFinp=21
          NFpMC=22

          NFout=31
          NFout2=32
          NFout3=33
          NFout4=34
          NFout5=35
          NFout6=36
          NFout7=37
          NFout8=38
          NFout9=39
          NFout0=40

          NFout2a=42
          NFout7a=47
          NFoutjt=50

        fileout='MCoutf2'
        INQUIRE (FILE= fileout,EXIST=zexist)
        if (.not.zexist) then
            OPEN (NFout2,STATUS='NEW',FORM='FORMATTED',FILE=fileout)
           else
            OPEN (NFout2,STATUS='OLD',FORM='FORMATTED',FILE=fileout)
        endif
        REWIND NFout2
```

```
      fileout='MCoutf2a'
      INQUIRE (FILE= fileout,EXIST=zexist)
      if (.not.zexist) then
           OPEN (NFout2a,STATUS='NEW',FORM='FORMATTED',FILE=fileout)
          else
           OPEN (NFout2a,STATUS='OLD',FORM='FORMATTED',FILE=fileout)
      endif
      REWIND NFout2a

      fileout='MCoutf3'
      INQUIRE (FILE= fileout,EXIST=zexist)
      if (.not.zexist) then
           OPEN (NFout3,STATUS='NEW',FORM='FORMATTED',FILE=fileout)
          else
           OPEN (NFout3,STATUS='OLD',FORM='FORMATTED',FILE=fileout)
      endif
      REWIND NFout3

      fileout='MCoutf4'
      INQUIRE (FILE= fileout,EXIST=zexist)
      if (.not.zexist) then
           OPEN (NFout4,STATUS='NEW',FORM='FORMATTED',FILE=fileout)
          else
           OPEN (NFout4,STATUS='OLD',FORM='FORMATTED',FILE=fileout)
      endif
      REWIND NFout4

      fileout='MCoutf5'
      INQUIRE (FILE= fileout,EXIST=zexist)
      if (.not.zexist) then
           OPEN (NFout5,STATUS='NEW',FORM='FORMATTED',FILE=fileout)
          else
           OPEN (NFout5,STATUS='OLD',FORM='FORMATTED',FILE=fileout)
      endif
      REWIND NFout5

      fileout='MCoutf6'
      INQUIRE (FILE= fileout,EXIST=zexist)
      if (.not.zexist) then
           OPEN (NFout6,STATUS='NEW',FORM='FORMATTED',FILE=fileout)
          else
           OPEN (NFout6,STATUS='OLD',FORM='FORMATTED',FILE=fileout)
      endif
      REWIND NFout6

      fileout='MCoutf7'
      INQUIRE (FILE= fileout,EXIST=zexist)
      if (.not.zexist) then
           OPEN (NFout7,STATUS='NEW',FORM='FORMATTED',FILE=fileout)
          else
           OPEN (NFout7,STATUS='OLD',FORM='FORMATTED',FILE=fileout)
      endif
      REWIND NFout7

      fileout='MCoutf7a'
      INQUIRE (FILE= fileout,EXIST=zexist)
      if (.not.zexist) then
           OPEN (NFout7a,STATUS='NEW',FORM='FORMATTED',FILE=fileout)
          else
           OPEN (NFout7a,STATUS='OLD',FORM='FORMATTED',FILE=fileout)
      endif
      REWIND NFout7a

      fileout='MCoutf8'
      INQUIRE (FILE= fileout,EXIST=zexist)
      if (.not.zexist) then
           OPEN (NFout8,STATUS='NEW',FORM='FORMATTED',FILE=fileout)
          else
           OPEN (NFout8,STATUS='OLD',FORM='FORMATTED',FILE=fileout)
      endif
      REWIND NFout8

      fileout='MCoutf9'
      INQUIRE (FILE= fileout,EXIST=zexist)
      if (.not.zexist) then
           OPEN (NFout9,STATUS='NEW',FORM='FORMATTED',FILE=fileout)
          else
           OPEN (NFout9,STATUS='OLD',FORM='FORMATTED',FILE=fileout)
      endif
      REWIND NFout9

      fileout='MCoutf0'
      INQUIRE (FILE= fileout,EXIST=zexist)
      if (.not.zexist) then
           OPEN (NFout0,STATUS='NEW',FORM='FORMATTED',FILE=fileout)
          else
           OPEN (NFout0,STATUS='OLD',FORM='FORMATTED',FILE=fileout)
      endif
      REWIND NFout0



      fileout='MCoutfjt'
      INQUIRE (FILE=fileout, EXIST=zexist)
      if (.not.zexist) then
        OPEN (NFoutjt, STATUS='NEW', FORM='FORMATTED', FILE=fileout)
       else
        OPEN (NFoutjt, STATUS='OLD', FORM='FORMATTED', FILE=fileout)
      endif
      REWIND NFoutjt


c  read the device configuration, bias and initial DD solution.
c  (use nmax as initial value of M; MCfDD reads a new value for M)

        M=nmax
        call MCfDD (tiphbt,matc,matb,mate,fmc,fmb,fme,tempK,M,ipi,
     +     ibt,imi,fet,t,psi0,efer,par,w,wi,nie,refp,nbmax,
     +     ptt,pttc,ni,s0,as,aco,aba,area,rc,rb,re,tic,
     +     mnhbt,mphbt,Ecrie,vsate,Ecrih,vsath,NFDD)

        if (M.gt.nmax) stop 'Device with more mesh points than expected'


c  Normalization parameters

cc (!!!) hereafter, tempK is assumed to be 300.0 Kelvin  (!!!)
cc      Vtnorm = bk * tempK
cc   If not, all the normalization parameters should be checked !!!

        parNorm(1) = Cnorm
      parNorm(2) = Dnorm
      parNorm(3) = Vtnorm
      parNorm(4) = tnorm
      parNorm(5) = xnorm
      parNorm(6) = vnorm
      parNorm(7) = Enorm
      parNorm(8) = knorm
      parNorm(9) = Fnorm
      parNorm(10)= Alfnorm
      parNorm(11)= jnorm

      parNorm(20) = He
      parNorm(21) = Hk
      parNorm(22) = Hv
      parNorm(23) = Hke

      parNorm(30) = pi
      parNorm(31) = q
      parNorm(32) = hbar
      parNorm(33) = bk
      parNorm(34) = eps0
      parNorm(35) = efm0

      rdx=base
      do i=0,iwmax
        wght(i)= rdx**i
      enddo

c  Some made assumptions and some needed param. for subroutines Poisson()
c   and DDholes():
c
c  Assumed constant values for some matrices defined and used in HBTsim,
c   that have been suppressed in this simulator:
c
c    - Region domain where the equations are solved
c       IPB(1)=1
c       IPB(2)=M
c
c    - Interface-related parameters at the heterojunctions
c       (surface charge density, trap recombination levels,
c        surface recombination velocity, ... )
c       PARINT(I,J)=0.0          PARINT(5,2)
c
c  Variables read-in in input():
c       ITERM,LPROX,CORL

      do 1 i=1,5
        PARINT(i,1)=0.0
        PARINT(i,2)=0.0
1     continue

c  read additional input data for the MC simulation

      call input (iseedc,iseeds,dt,jTh,jTm1,jTl1,jTm2,jTl2,
     +            jTm3,jTl3,jdots,xlow,xhigh,de,iemax,epp,NFinp,
     +            jfrequen,ITERM,LPROX,CORL)

c  normalization of (the recently introduced) variables

      dt=dt/tnorm
      de=de/Enorm
      de1=1.0/de
      xlow=xlow/xnorm
      xhigh=xhigh/xnorm

      do i=1,3
        Ecrie(i)=Ecrie(i)/Fnorm
        vsate(i)=vsate(i)/vnorm
        Ecrih(i)=Ecrih(i)/Fnorm
        vsath(i)=vsath(i)/vnorm
      enddo

c  compute the carrier mobilities according to the present electric field

      call field (gradp,w,par,ipi,M,nbmax)

      if (tiphbt .eq. 2)      call muofe (ipi,par,gradp,M,nmax,
     +                          mnhbt,mphbt,Ecrie,vsate,Ecrih,vsath)

c  store the initial solutions

      call MCoutput2 (M,par,parNorm,Lnorm,w,gradp,nbmax,NFout2)
      call MCoutput3 (M,par,parNorm,Lnorm,w,nie,npti,cn,nbmax,NFout3)

c  initialize the random number generator

      do 2 i=1,nthds
        call rcarin (iseedc-i,rnc(0,i),0)
        call rcarin (iseeds+i,rns(0,i),0)
        call rcarry (rnc(0,i),5)
        call rcarry (rns(0,i),Lrns*Lbck)
        La(1,i)=0
2     continue

c   MC parameters and initial conditions for the MC dynamics
c       [ also calculated: par(18,j) and par(19,j) ]

      call paramMC (tiphbt,matc,matb,mate,fmc,fmb,fme,tempK,
     +        LNorm,parNorm,M,ipi,par,ivmax,iregnmax,ismmax,ienmax,
     +        de,iemax,efm,alf,gm,swk,ec,hwo,hwij,hwe,NFpMC)

      call initiaMC(nptmax,npt,pt,ipt,rnc,epp,
     +        npcmax,npcmin,iwmax,ivmax,iregnmax,efm,alf,gm,
     +        LNorm,parNorm,M,par,nie,w,nbmax,ipi)

      cn0c= nie(1,1)*exp(w(1,1)-w(3,1))
      cn0e= nie(1,M)*exp(w(1,M)-w(3,M))

c  Initial value of the recombination current density
      jcol=10000
      write(*,*) 'Recombination current according to DD_HBTSIM:'
      call densi (M,W,PAR,IPI,IBT,NBMAX,NMAX,NIE,wi,PARINT,NFoutjt,
     +   IMI,PTT,FET,PTTC,jcol,VBE,AREA,T,NI,S0,AS,RC,RB,RE,ABA,ACO)


      call recomb (M,IPI,IMI,IBT,
     +        T,W,WI,PAR,PARINT,NIE,NBMAX,IR,IRCOL,IRTOTAL)
c     print*, IRCOL, IRTOTAL

      write(*,*)'Recombinacio entre terminal C i B :   ',
     +                  IRCOL*JNORM
      write(*,*)'Recombinacio entre terminal E i B :   ',
     +                   (IRTOTAL-IRCOL)*JNORM
        write(*,*)'Recombinacio total = corrent base :    ',
     +                   IRTOTAL*JNORM
        write(*,*)
        write(*,*)


c   Transmission coefficient for electrons near the heterojunction

        jl =ipi(2)
        j=2
3       if ( par(ixm,jl-j) .ge. par(ixm,jl)-100 ) then
          j=j+1
          goto 3
        endif
        jl =ipi(2) - j

      do iv=1,ivmax
        jh(iv) = ipi(2) + 10
      enddo

      if (mate.ne.matb) call tunnel (Teb,deT,w,par,ec,efm,alf,E0,refp,
     +  Alfnorm,Enorm,xnorm,M,ipi,ibt,nbmax,ivmax,iregnmax,ieTmax,jh)


c   Self-consistent MC-Poisson iteration.

        time=0.0

        znqj = .false.
        frepe = 0

        lfrtop=0
        npttop=npt
        nptitop=(npcmin+npcmax)/2

c       call MCoutput2 (M,par,parNorm,Lnorm,w,gradp,nbmax,NFout2)
c       call MCoutput3 (M,par,parNorm,Lnorm,w,nie,npti,cn,nbmax,NFout3)
        call MCoutput4 (par,parNorm,ipi,w,refp,pt,ipt,ec,cn,vxm,fdl,
     +   npt,time,alf,efm,epp,M,LNorm,nbmax,ivmax,iregnmax,
     +   nptmax,cn2,iwmax,wght,0,xlow,xhigh,NFout4,NFout5,NFout6)

      if (mate.ne.matb) then
        do i=1,ieTmax/4
          write(NFout9,900)      Enorm*(E0(1)+(i-1)*deT),Teb(i,1),
     +                           Enorm*(E0(2)+(i-1)*deT),Teb(i,2)
        enddo
        write(NFout9,901) ''
        write(NFout0,900)
     +          (par(ixm,jl)    -par(ixm,ipi(2)))*xnorm*1.0d+08,
     +          (par(ixm,jh(1))-par(ixm,ipi(2)))*xnorm*1.0d+08,
     +          (par(ixm,jh(2))-par(ixm,ipi(2)))*xnorm*1.0d+08
      else
          write(NFout9,*) 'Transmission coeff not computed (device=BJT)'
      endif


c  Restarting point in order to compute the small signal parameters
5       continue
```

```
        do 10 jT=1,jTh

        do 12 lll=1,jTll*jTml

        time=time+dt

        call MCdinam (time,dt,npt,pt,ipt,rnc,rns,Lrns,Lbck,L2i,nthds,
     +          LB,La,ivmax,iregnmax,ismmax,ienmax,iemax,nptmax,
     +          del,efm,alf,gm,swk,ec,hwo,hwij,hwe,
     +          refp,E0,Teb,deT,ieTmax,matb,mate,jl,jh,ibt,
     +          nqj,nqjaux,nqjt,nqjtaux,znqj,wght,iwmax,nmax,
     +          nbmax,M,gradp,w,nie,par,parNorm,LNorm,ipi)

        call chargeCIC(pt,ipt,npt,nptmax,time+dt,epp,cn0c,cn0e,cn,
     +          npti,nptiw,nptnw,ifree,ipi,w,par,M,efm,alf,gm,rnc,
     +          cnaux,LB,nptcon,nptcon4,joutc,joute,joutcaux,jouteaux,
     +          nptiwaux,nptnwaux,jelim,wght,
     +          nmax,iwmax,inmax,lfrmax,lfrtop,npttop,nptitop,nthds,
     +          parNorm,LNorm,nbmax,ivmax,iregnmax,npcmax,npcmin)

        call DDholes (PARINT,T,IPI,IMI,IBT,
     +          M,W,PAR,NIE,S,B,C,MP,NBMAX,NMAX,pdt0,dt,
     +          NTMAX,ITERM,LPROX,ERROR,CORRE,ITER,CORL,TIP,wi,fet)

        if (TIP.ne.1) then
                write (*,*) jT,time/dt,time,error,corre,iter
                stop 'No convergence in DDholes()'
        endif

        call Poisson (psi0,PARINT,t,ipi,M,w,par,nie,S,B,C,MP,nbmax,
     +          nmax,ntmax,ITERM,LPROX,ERROR,CORRE,ITER,CORL,TIP)

        if (TIP.ne.1) then
                write (*,*) jT,time/dt,error,corre,iter
                stop 'No convergence in Poisson()'
        endif


        call field (gradp,w,par,ipi,M,nbmax)

        if (tiphbt .eq. 2)      call muofe (ipi,par,gradp,M,nmax,
     +                            mnhbt,mphbt,Ecrie,vsate,Ecrih,vsath)

      if (mate.ne.matb) call tunnel (Teb,deT,w,par,ec,efm,alf,E0,refp,
     +  Alfnorm,Enorm,xnorm,M,ipi,ibt,nbmax,ivmax,iregnmax,ieTmax,jh)


12      continue

        do 16 mmm=1,jTm2
        do 15 lll=1,jTl2

        time=time+dt

        call MCdinam (time,dt,npt,pt,ipt,rnc,rns,Lrns,Lbck,L2i,nthds,
     +          LB,La,ivmax,iregnmax,ismmax,ienmax,iemax,nptmax,
     +          del,efm,alf,gm,swk,ec,hwo,hwij,hwe,
     +          refp,E0,Teb,deT,ieTmax,matb,mate,jl,jh,ibt,
     +          nqj,nqjaux,nqjt,nqjtaux,znqj,wght,iwmax,nmax,
     +          nbmax,M,gradp,w,nie,par,parNorm,LNorm,ipi)
```

```
      call chargeCIC(pt,ipt,npt,nptmax,time+dt,epp,cn0c,cn0e,cn,
     +        npti,nptiw,nptnw,ifree,ipi,w,par,M,efm,alf,gm,rnc,
     +        cnaux,LB,nptcon,nptcon4,joutc,joute,joutcaux,jouteaux,
     +        nptiwaux,nptnwaux,jelim,wght,
     +        nmax,iwmax,inmax,lfrmax,lfrtop,npttop,nptitop,nthds,
     +        parNorm,LNorm,nbmax,ivmax,iregnmax,npcmax,npcmin)

      call DDholes (PARINT,T,IPI,IMI,IBT,
     +        M,W,PAR,NIE,S,B,C,MP,NBMAX,NMAX,pdt0,dt,
     +        NTMAX,ITERM,LPROX,ERROR,CORRE,ITER,CORL,TIP,wi,fet)

      if (TIP.ne.1) then
              write (*,*) jT,time/dt,time,error,corre,iter
              stop 'No convergence in DDholes()'
      endif

      call Poisson (psi0,PARINT,t,ipi,M,w,par,nie,S,B,C,MP,nbmax,
     +        nmax,ntmax,ITERM,LPROX,ERROR,CORRE,ITER,CORL,TIP)

      if (TIP.ne.1) then
              write (*,*) jT,time/dt,error,corre,iter
              stop 'No convergence in Poisson()'
      endif


      call field (gradp,w,par,ipi,M,nbmax)

      if (tiphbt .eq. 2)      call muofe (ipi,par,gradp,M,nmax,
     +                          mnhbt,mphbt,Ecrie,vsate,Ecrih,vsath)


     if (mate.ne.matb) call tunnel (Teb,deT,w,par,ec,efm,alf,E0,refp,
     +  Alfnorm,Enorm,xnorm,M,ipi,ibt,nbmax,ivmax,iregnmax,ieTmax,jh)

15    continue

      call MCoutput2 (M,par,parNorm,Lnorm,w,gradp,nbmax,NFout2)
      call MCoutput3 (M,par,parNorm,Lnorm,w,nie,npti,cn,nbmax,NFout3)
      call MCoutput4 (par,parNorm,ipi,w,refp,pt,ipt,ec,cn,vxm,fd1,
     +  npt,time,alf,efm,epp,M,LNorm,nbmax,ivmax,iregnmax,
     +  nptmax,cn2,iwmax,wght,0,xlow,xhigh,NFout4,NFout5,NFout6)

     if (mate.ne.matb) then
      do i=1,ieTmax/4
        write(NFout9,900)      Enorm*(E0(1)+(i-1)*deT),Teb(i,1),
     +                         Enorm*(E0(2)+(i-1)*deT),Teb(i,2)
      enddo
      write(NFout9,901) ''
      write(NFout0,900)
     +        (par(ixm,jl)   -par(ixm,ipi(2)))*xnorm*1.0d+08,
     +        (par(ixm,jh(1))-par(ixm,ipi(2)))*xnorm*1.0d+08,
     +        (par(ixm,jh(2))-par(ixm,ipi(2)))*xnorm*1.0d+08
     else
        write(NFout9,*) 'Transmission coeff not computed (device=BJT)'
     endif


      call recomb (M,IPI,IMI,IBT,
     +       T,W,WI,PAR,PARINT,NIE,NBMAX,IR,IRCOL,IRTOTAL)
c     print*, IRCOL, IRTOTAL
```

```
      write(*,*)'Recombinacio entre terminal C i B :   ',
     +                IRCOL*JNORM
      write(*,*)'Recombinacio entre terminal E i B :   ',
     +                (IRTOTAL-IRCOL)*JNORM
      write(*,*)'Recombinacio total = corrent base :   ',
     +                IRTOTAL*JNORM
      write(*,*)
      write(*,*)



16    continue


10    continue


c   Time averaging to compute statistical quantities

     do 17 j=1,M
      Scn(j)= 0.0
      Scn2(j)= 0.0
      Svx(j)= 0.0
      Sek(1,j)= 0.0
      Sek(2,j)= 0.0
      Sfd1(j)= 0.0
17   continue

      do j=1,M
        Sw(1,j)  = 0.0
        Sw(2,j)  = 0.0
        Sw(3,j)  = 0.0
      enddo

      znqj = .true.
      do iv=1,ivmax
       nqjt(1,iv)=0
       nqjt(2,iv)=0
       nqjt(3,iv)=0
       nqjt(4,iv)=0
       do j=1,M
              nqj(j,iv) = 0
       enddo
      enddo

      do 19 mmm=1,jTm3
      do 18 lll=1,jTl3

      time=time+dt

      call MCdinam (time,dt,npt,pt,ipt,rnc,rns,Lrns,Lbck,L2i,nthds,
     +        LB,La,ivmax,iregnmax,ismmax,ienmax,iemax,nptmax,
     +        del,efm,alf,gm,swk,ec,hwo,hwij,hwe,
     +        refp,E0,Teb,deT,ieTmax,matb,mate,jl,jh,ibt,
     +        nqj,nqjaux,nqjt,nqjtaux,znqj,wght,iwmax,nmax,
     +        nbmax,M,gradp,w,nie,par,parNorm,LNorm,ipi)

      call chargeCIC(pt,ipt,npt,nptmax,time+dt,epp,cn0c,cn0e,cn,
     +        npti,nptiw,nptnw,ifree,ipi,w,par,M,efm,alf,gm,rnc,
     +        cnaux,LB,nptcon,nptcon4,joutc,joute,joutcaux,jouteaux,
     +        nptiwaux,nptnwaux,jelim,wght,
     +          nmax,iwmax,inmax,lfrmax,lfrtop,npttop,nptitop,nthds,
     +          parNorm,LNorm,nbmax,ivmax,iregnmax,npcmax,npcmin)

        call DDholes (PARINT,T,IPI,IMI,IBT,
     +          M,W,PAR,NIE,S,B,C,MP,NBMAX,NMAX,pdt0,dt,
     +          NTMAX,ITERM,LPROX,ERROR,CORRE,ITER,CORL,TIP,wi,fet)

        if (TIP.ne.1) then
                write (*,*) jT,time/dt,time,error,corre,iter
                stop 'No convergence in DDholes()'
        endif

        call Poisson (psi0,PARINT,t,ipi,M,w,par,nie,S,B,C,MP,nbmax,
     +          nmax,ntmax,ITERM,LPROX,ERROR,CORRE,ITER,CORL,TIP)

        if (TIP.ne.1) then
                write (*,*) jT,time/dt,error,corre,iter
                stop 'No convergence in Poisson()'
        endif


        call field (gradp,w,par,ipi,M,nbmax)

        if (tiphbt .eq. 2)      call muofe (ipi,par,gradp,M,nmax,
     +                            mnhbt,mphbt,Ecrie,vsate,Ecrih,vsath)


      if (mate.ne.matb) call tunnel (Teb,deT,w,par,ec,efm,alf,E0,refp,
     +  Alfnorm,Enorm,xnorm,M,ipi,ibt,nbmax,ivmax,iregnmax,ieTmax,jh)

18      continue

cc    Sample signals and accumulate
cc       xi = ...
cc       Sxi = Sxi + xi

        call Accumul (pt,ipt,npt,nptmax,epp,cn,vxm,ekm,fd1,
     +          Scn,Svx,Sek,Sfd1,cnaux,vxmaux,ekmaux,fd1aux,ipi,
     +          nmax,nthdsmax,nthds,LB,npti,par,M,efm,alf,gm,
     +          cn2,cn2aux,Scn2,iwmax,wght,parNorm,LNorm,ivmax,iregnmax)

        do j=1,M
          Sw(1,j) = Sw(1,j) + w(1,j)
          Sw(2,j) = Sw(2,j) + w(2,j)
          Sw(3,j) = Sw(3,j) + w(3,j)
        enddo


19      continue

cc    Compute mean values and record results to a file
cc       Sxi = Sxi /jTm3
cc       write () Sxi

      if (jTm3.gt.0) then

        do j=1,M
          Sw(1,j) = Sw(1,j) /jTm3
          Sw(2,j) = Sw(2,j) /jTm3
          Sw(3,j) = Sw(3,j) /jTm3
        enddo

        call field (gradp,Sw,par,ipi,M,nbmax)

        do j=1,M
          write(NFout2a,900)
     +          par(ixm,j)*xnorm*1.0d+08,
     +          Sw(1,j)*Vtnorm, Sw(2,j)*Vtnorm,
     +          Sw(3,j)*Vtnorm,-gradp(j)*Fnorm*1.0d-03
        enddo

       do 200 j=1,M
         Svx(j) = Svx(j) *vnorm / Scn(j)
         Sek(1,j) = Sek(1,j) *Enorm / Sfd1(j)
         if ( Scn(j) .le. Sfd1(j)+MP ) then
            if (j.gt.1) then
              Sek(2,j) = Sek(2,j-1)
            else
              Sek(2,j) = 0.0
            endif
         else
              Sek(2,j) = Sek(2,j) *Enorm / ( Scn(j) - Sfd1(j) )
         endif
         Sfd1(j)= Sfd1(j) / Scn(j)
         Scn(j) = Scn(j) *epp/ (xnorm*1.0d+4*par(idx,j)*jTm3)
200    continue

c  Correction related to the CIC method (half box integration; triangle shape):
c!!     (fd1() should be calculated too)

c       j=ipi(1)
c       Scn(j)=Scn(j)+0.5*(Scn(j)-Scn(j-1))
c       Svx(j)=Svx(j)+0.5*(Svx(j)-Svx(j-1))
c       Sek(1,j)=Sek(1,j)+0.5*(Sek(1,j)-Sek(1,j-1))
c       Sek(2,j)=Sek(2,j)+0.5*(Sek(2,j)-Sek(2,j-1))
c
c       j=ipi(1)+1
c       Scn(j)=Scn(j)-0.5*(Scn(j+1)-Scn(j))
c       Svx(j)=Svx(j)-0.5*(Svx(j+1)-Svx(j))
c       Sek(1,j)=Sek(1,j)-0.5*(Sek(1,j+1)-Sek(1,j))
c       Sek(2,j)=Sek(2,j)-0.5*(Sek(2,j+1)-Sek(2,j))

c !!! Only for homojunctions in the BC interface

        j=ipi(1)
        Scn(j)=0.5*(Scn(j)+Scn(j+1))
        Svx(j)=0.5*(Svx(j)+Svx(j+1))
        Sek(1,j)=0.5*(Sek(1,j)+Sek(1,j+1))
        Sek(2,j)=0.5*(Sek(2,j)+Sek(2,j+1))

        Scn(j+1)=Scn(j)
        Svx(j+1)=Svx(j)
        Sek(1,j+1)=Sek(1,j)
        Sek(2,j+1)=Sek(2,j)


c Valid for homo- & heterojunctions in the EB interface

        j=ipi(2)
        Scn(j)=Scn(j)+0.5*(Scn(j)-Scn(j-1))
        Svx(j)=Svx(j)+0.5*(Svx(j)-Svx(j-1))
        Sek(1,j)=Sek(1,j)+0.5*(Sek(1,j)-Sek(1,j-1))
        Sek(2,j)=Sek(2,j)+0.5*(Sek(2,j)-Sek(2,j-1))

        j=ipi(2)+1
        Scn(j)=Scn(j)-0.5*(Scn(j+1)-Scn(j))
        Svx(j)=Svx(j)-0.5*(Svx(j+1)-Svx(j))
        Sek(1,j)=Sek(1,j)-0.5*(Sek(1,j+1)-Sek(1,j))
        Sek(2,j)=Sek(2,j)-0.5*(Sek(2,j+1)-Sek(2,j))


      if (frepe .eq. 0) then
        do 300 j=1,M
          write(NFout7,900)
     +          par(ixm,j)*xnorm*1.0d+08,
     +          Scn(j), -Svx(j),
     +          -q*Scn(j)*Svx(j), 1.0-Sfd1(j)

          write(NFout8,900)
     +          par(ixm,j)*xnorm*1.0d+08,
     +          Sek(1,j)*Sfd1(j) + Sek(2,j)*(1.0-Sfd1(j)),
     +          Sek(1,j), Sek(2,j)
 300    continue

        factorj = -q *epp/(jTm3*jTl3*dt*tnorm*1.0d+04)

        do j=1,M
          write(NFout7a,900)
     +          par(ixm,j)*xnorm*1.0d+08,
     +          nqj(j,1)*factorj    ,
     +          (nqj(j,1)+nqj(j,2))*factorj
        enddo

        print*, ''
        print*, 'Thermionic and tunneling currents at the E-B junction:'
        print*, '          Jth_E-->B   Jth_B-->E   Jtu_E-->B   Jtu_B-->E'
        print*, 'iv=1:'
        print*, '          Jth_E-->B = ',  nqjt(1,1)*factorj
        print*, '          Jth_B-->E = ',  nqjt(2,1)*factorj
        print*, '          Jtu_E-->B = ',  nqjt(3,1)*factorj
        print*, '          Jtu_B-->E = ',  nqjt(4,1)*factorj
        print*, ''

        print*, 'iv=2:'
        print*, '          Jth_E-->B = ',  nqjt(1,2)*factorj
        print*, '          Jth_B-->E = ',  nqjt(2,2)*factorj
        print*, '          Jtu_E-->B = ',  nqjt(3,2)*factorj
        print*, '          Jtu_B-->E = ',  nqjt(4,2)*factorj
        print*, ''

        n=0
        do iv=1,ivmax
         do i=1,4
           n=n+ nqjt(i,iv)
         enddo
        enddo

        print*, 'Total:',
     +          n*factorj

        print*, ''
        print*, ''
        endif
        endif

        if (frepe .eq. 0) then
          print*, 'Array out-of-range indexing test:'
          print*,'         lfrtop,       npttop,       nptitop  '
          print*,          lfrtop,       npttop,       nptitop
          print*, ''

         if (jdots.eq.1) then
          REWIND NFout5
          call MCoutput4 (par,parNorm,ipi,w,refp,pt,ipt,ec,cn,vxm,fd1,
     +     npt,time,alf,efm,epp,M,LNorm,nbmax,ivmax,iregnmax,
     +     nptmax,cn2,iwmax,wght,jdots,xlow,xhigh,NFout4,NFout5,NFout6)
         endif

c         call MCoutput (tiphbt,matc,matb,mate,fmc,fmb,fme,tempK,M,
c     +        ipi,ibt,imi,fet,t,psi0,efer,par,w,wi,nie,nbmax,NFout)

        if (mate.ne.matb) then
          do i=1,ieTmax/4
            write(NFout9,900)      Enorm*(E0(1)+(i-1)*deT),Teb(i,1),
     +                             Enorm*(E0(2)+(i-1)*deT),Teb(i,2)
          enddo
          write(NFout9,901) ''
          write(NFout0,900)
     +           (par(ixm,jl)    -par(ixm,ipi(2)))*xnorm*1.0d+08,
     +           (par(ixm,jh(1))-par(ixm,ipi(2)))*xnorm*1.0d+08,
     +           (par(ixm,jh(2))-par(ixm,ipi(2)))*xnorm*1.0d+08
        else
            write(NFout9,*) 'Transmission coeff not computed (device=BJT)'
        endif
        endif



        if (jTm3.gt.0) then

          DO J=1,M
           FNC = (Sw(1,J)-Sw(3,J)) + PAR(16,J)
           FNV = (Sw(2,J)-Sw(1,J)) + PAR(17,J)
           NIE(1,J) = PAR(13,J)*FERM12(FNC)/EXP(FNC)
           NIE(2,J) = PAR(14,J)*FERM12(FNV)/EXP(FNV)
          ENDDO

          jelec = 0.0
          do j=ipi(1),ipi(2)-2
            jelec = jelec + Scn(j)*Svx(j)
          enddo
          jelec=-q*jelec/float(ipi(2)-1-ipi(1))

        if (frepe .eq. 0) then
         jcol=jelec
         call densi (M,Sw,PAR,IPI,IBT,NBMAX,NMAX,NIE,wi,PARINT,NFoutjt,
     +     IMI,PTT,FET,PTTC,jcol,VBE,AREA,T,NI,S0,AS,RC,RB,RE,ABA,ACO)

         if (jfrequen .eq. 1) then
         call frequen (PAR,NIE,Sw,IPI,M,FREPE,NMAX,NBMAX,
     +       TIC,AREA,IBT,XCELE,XCHUE,jcol,jelec)
```

```
         t(1) = t(1) + tic/Enorm
         t(2) = t(2) + tic/Enorm
          call Poisson (psi0,PARINT,t,ipi,M,w,par,nie,S,B,C,MP,nbmax,
     +          nmax,ntmax,ITERM,LPROX,ERROR,CORRE,ITER,CORL,TIP)
          call field (gradp,w,par,ipi,M,nbmax)
         frepe = 1
         znqj = .false.
         jTl1=(jTh*(jTm1*jTl1+jTm2*jTl2))/10
         jTh=1
         jTm1=1
         jTm2=0
         jTl2=0
         goto 5
         endif
        else
         call frequen (PAR,NIE,Sw,IPI,M,FREPE,NMAX,NBMAX,
     +       TIC,AREA,IBT,XCELE,XCHUE,jcol,jelec)
        endif


        else


          factorj = -q*epp*vnorm/(1.0d+4*xnorm)
          jelec = 0.0
          do j=ipi(1),ipi(2)-2
            jelec = jelec + cn(j)*vxm(j)*factorj/par(idx,j)
          enddo
          jelec=-q*jelec/float(ipi(2)-1-ipi(1))

        if (frepe .eq. 0) then
         jcol=jelec
         call densi (M,W,PAR,IPI,IBT,NBMAX,NMAX,NIE,wi,PARINT,NFoutjt,
     +     IMI,PTT,FET,PTTC,jcol,VBE,AREA,T,NI,S0,AS,RC,RB,RE,ABA,ACO)

         if (jfrequen .eq. 1) then
         call frequen (PAR,NIE,W,IPI,M,FREPE,NMAX,NBMAX,
     +       TIC,AREA,IBT,XCELE,XCHUE,jcol,jelec)

         t(1) = t(1) + tic/Enorm
         t(2) = t(2) + tic/Enorm
          call Poisson (psi0,PARINT,t,ipi,M,w,par,nie,S,B,C,MP,nbmax,
     +          nmax,ntmax,ITERM,LPROX,ERROR,CORRE,ITER,CORL,TIP)
          call field (gradp,w,par,ipi,M,nbmax)
         frepe = 1
         znqj = .false.
         jTl1=(jTh*(jTm1*jTl1+jTm2*jTl2))/10
         jTh=1
         jTm1=1
         jTm2=0
         jTl2=0
         goto 5
         endif
        else
         call frequen (PAR,NIE,W,IPI,M,FREPE,NMAX,NBMAX,
     +       TIC,AREA,IBT,XCELE,XCHUE,jcol,jelec)
        endif

        endif
```

```
      close (NFout2)
      close (NFout2a)
      close (NFout3)
      close (NFout4)
      close (NFout5)
      close (NFout6)
      close (NFout7)
      close (NFout7a)
      close (NFout8)
      close (NFout9)
      close (NFout0)
      close (NFoutjt)


900    FORMAT (T1,5(G14.7,2X))
901    FORMAT (A)

      stop
      end

c************************ subroutine MCfDD ***************************
c---------------------------------------------------------------------
c
c Subroutine to read the standard input data file of the MCHBT simulator
c
c  This subroutine is to be executed by the MCHBT simulator in order
c to read in an initial solution provided by any DD simulator.
c
c
c  The requested information is:
c       - HBT type (abrupts only); CBE material; CBE molar fraction
c       - number of grid points
c       - CBE grid indices
c       - some physical parameters (at each grid point) used in DD model
c       - CBE bias; T(K)
c       - equilibrium potential at C and E contacts
c       - HBTsim solution for potentials (at each grid point)
c         (quasi-Fermi voltage-levels for electrons and holes are both
c          assumed to be identically zero at equilibrium
c          [voltage reference])
c
c                   by Pau Garcias i Salva'. July, 1997
c
c---------------------------------------------------------------------
       subroutine MCfDD (tiphbt,matc,matb,mate,fmc,fmb,fme,tempK,M,
     +     ipi,ibt,imi,fet,t,psi0,efer,par,w,wi,nie,refp,nbmax,
     +     ptt,pttc,ni,s0,as,aco,aba,area,rc,rb,re,tic,
     +     mnhbt,mphbt,Ecrie,vsate,Ecrih,vsath,nf)
c
c      storage allocation
c
c           implicit double precision (a-h,o-z)
c           implicit integer (i-n)
       implicit none
c
           integer
     +         tiphbt,matc,matb,mate,M,ipi(2),ibt,imi(2),nbmax
           double precision
     +         fmc,fmb,fme,tempK,t(3),psi0(2),efer,par(19,0:M+1),
     +         w(nbmax,M),wi(1,M),nie(2,M),fet(5,4),refp(5),
     +         ptt,pttc,ni,s0,as,aco,aba,area,rc,rb,re,tic,
     +         mnhbt(M),mphbt(M),
     +         Ecrie(3),vsate(3),Ecrih(3),vsath(3)
           integer
     +         nf

           integer
     +         i,j
           logical
     +         zexist
           character*16
     +         filein



      filein='DD2MC.dat'
      OPEN (nf,STATUS='OLD',FORM='FORMATTED',FILE=filein,ERR=99)
      REWIND nf


        read(nf,90) tiphbt
        read(nf,90) matc,matb,mate
        read(nf,91) fmc,fmb,fme
        read(nf,90) M
        read(nf,90) ipi(1),ibt,ipi(2)
        read(nf,90) imi(1),imi(2)

        read(nf,*)
        read(nf,*)

        read(nf,91) refp(1),refp(2),refp(3)
        read(nf,91) refp(4),refp(5)

        read(nf,*)
        read(nf,*)

        read(nf,91) Ecrie(1),Ecrie(2),Ecrie(3)
        read(nf,91) vsate(1),vsate(2),vsate(3)
        read(nf,91) Ecrih(1),Ecrih(2),Ecrih(3)
        read(nf,91) vsath(1),vsath(2),vsath(3)

        read(nf,*)
        read(nf,*)

        do 10 j=1,M
          read(nf,90) i
          read(nf,91) par(1,i),par(2,i),par(3,i)
          read(nf,91) par(4,i),par(5,i),par(6,i)
          read(nf,91) par(7,i),par(8,i),par(9,i)
          read(nf,91) par(10,i),par(11,i),par(12,i)
          read(nf,91) par(13,i),par(14,i),par(15,i)
          read(nf,91) par(16,i),par(17,i)
          read(nf,91) mnhbt(i),mphbt(i)
          read(nf,*)
10      continue

        read(nf,*)
        read(nf,*)

        do 15 i=1,4
          read(nf,91) fet(1,i),fet(2,i),fet(3,i)
          read(nf,91) fet(4,i),fet(5,i)
15      continue

        read(nf,*)
        read(nf,*)

        read(nf,91) pttc,ptt
        read(nf,91) ni,s0,as
        read(nf,91) aco,aba,area
        read(nf,91) rc,rb,re
        read(nf,91) tic

        read(nf,*)
        read(nf,*)

        read(nf,91) t(1),t(2),t(3)
        read(nf,91) tempK
        read(nf,91) psi0(1),psi0(2),efer

        read(nf,*)
        read(nf,*)

        do 20 j=1,M
          read(nf,90) i
          read(nf,91) w(1,i),w(2,i),w(3,i)
          read(nf,91) wi(1,i),nie(2,i),nie(1,i)
          read(nf,*)
20      continue


      close (nf)
90    FORMAT (1X,3I5)
91    FORMAT (1X,3G23.16)


      return
99    INQUIRE (FILE= filein,EXIST=zexist)
      if (.not.zexist) then
           print*, filein,': File not found'
        else
           print*,'Error while opening file'
      endif
c
      stop 'Error trying to open/read input data file'

c      return
      end


c************************ subroutine muofe **************************
c-------------------------------------------------------------------
c       Carrier mobility as a function of the electric field.
c
c                   by Pau Garcias i Salva'. Feb., 1999
c
c-------------------------------------------------------------------
       subroutine muofe (ipi,par,gradp,M,nmax,
     +                 mnhbt,mphbt,Ecrie,vsate,Ecrih,vsath)
```

```
c
c       storage allocation
c
c            implicit double precision (a-h,o-z)
c            implicit integer (i-n)
        implicit none
c
            integer
     +          ipi(2),M,nmax

            double precision
     +          gradp(0:nmax+1),par(19,0:nmax+1),
     +          mnhbt(nmax),mphbt(nmax),
     +          Ecrie(3),vsate(3),Ecrih(3),vsath(3)

            integer
     +          i,j
            double precision
     +          En


        j=1
        do i=1,ipi(1)
          En=dabs(gradp(i))
          par(3,i)=(mnhbt(i)+vsate(j)/Ecrie(j)*(En/Ecrie(j))**3)/
     +          (1+(En/Ecrie(j))**4)
          par(4,i)=(mphbt(i)+vsath(j)/Ecrih(j)*(En/Ecrih(j))**3)/
     +          (1+(En/Ecrih(j))**4)
        enddo

        j=2
        do i=ipi(1)+1,ipi(2)
          En=dabs(gradp(i))
          par(3,i)=(mnhbt(i)+vsate(j)/Ecrie(j)*(En/Ecrie(j))**3)/
     +          (1+(En/Ecrie(j))**4)
          par(4,i)=(mphbt(i)+vsath(j)/Ecrih(j)*(En/Ecrih(j))**3)/
     +          (1+(En/Ecrih(j))**4)
        enddo

        j=3
        do i=ipi(2)+1,M
          En=dabs(gradp(i))
          par(3,i)=(mnhbt(i)+vsate(j)/Ecrie(j)*(En/Ecrie(j))**3)/
     +          (1+(En/Ecrie(j))**4)
          par(4,i)=(mphbt(i)+vsath(j)/Ecrih(j)*(En/Ecrih(j))**3)/
     +          (1+(En/Ecrih(j))**4)
        enddo


c      return
      end

ceeee+i**#**** ****#**** ****#**** ****#**** ****#**** ****#**** ****#72------80
```

```
ceeee+i**#**** ****#**** ****#**** ****#**** ****#**** ****#**** ****#72------80

c*********************** subroutine input **************************
c-----------------------------------------------------------------------
c
c
c                  by Pau Garcias i Salva'. Nov., 1997
c
c-----------------------------------------------------------------------
       subroutine input(iseedc,iseeds,dt,jTh,jTm1,jTl1,jTm2,jTl2,
     +        jTm3,jTl3,jdots,xlow,xhigh,de,iemax,epp,NF,
     +        jfrequen,ITERM,LPROX,CORL)
c
c      storage allocation
c
c           implicit double precision (a-h,o-z)
c           implicit integer (i-n)

       implicit none
c
           integer
     +         iseedc,iseeds,jTh,jTm1,jTl1,jTm2,jTl2,jTm3,jTl3,
     +         iemax,jdots,jfrequen
           double precision
     +         dt,de,epp,xlow,xhigh
           integer
     +         NF
           logical
     +         zexist
           character*16
     +         filein

           INTEGER
     +         ITERM
           DOUBLE PRECISION
     +         LPROX,CORL


      filein='inputMC.dat'
      OPEN (NF,STATUS='OLD',FORM='FORMATTED',FILE=filein,ERR=99)
      REWIND NF

       read(NF,*) iseedc
       read(NF,*) iseeds
       read(NF,*) dt
       read(NF,*) jTh
       read(NF,*) jTm1
       read(NF,*) jTl1
       read(NF,*) jTm2
       read(NF,*) jTl2
       read(NF,*) jTm3
       read(NF,*) jTl3
       read(NF,*) de
       read(NF,*) iemax
       read(NF,*) epp

       read(NF,*) ITERM
       read(NF,*) LPROX
       read(NF,*) CORL
       read(NF,*) jdots
       read(NF,*) xlow
       read(NF,*) xhigh

       read(NF,*) jfrequen

      close (NF)
90    FORMAT (1X,3I5)
91    FORMAT (1X,3G23.16)


      return
99    INQUIRE (FILE= filein,EXIST=zexist)
      if (.not.zexist) then
           print*, filein,': File not found'
        else
           print*,'Error while opening file'
      endif
c
      stop 'Error trying to open/read input data file'

c      return
      end
c*********************** subroutine paramMC ***********************
c-----------------------------------------------------------------------
c
c
c                  by Pau Garcias i Salva'. Nov., 1997
c
c-----------------------------------------------------------------------
       subroutine paramMC(tiphbt,matc,matb,mate,fmc,fmb,fme,tempK,
     +        LNorm,parNorm,M,ipi,par,ivmax,iregnmax,ismmax,ienmax,
     +        de,iemax,efm,alf,gm,swk,ec,hwo,hwij,hwe,NF)
c
c      storage allocation
c
c           implicit double precision (a-h,o-z)
c           implicit integer (i-n)

       implicit none
c
           integer
     +         tiphbt,matc,matb,mate
           double precision
     +         fmc,fmb,fme,tempK
           integer
     +         LNorm,M,ipi(2),ivmax,iregnmax,ismmax,ienmax,iemax
           double precision
     +         parNorm(LNorm),par(19,0:M+1),de,efm(ivmax,iregnmax),
     +         alf(ivmax,iregnmax),gm(ivmax,iregnmax),
     +         swk(ivmax,ismmax,ienmax,iregnmax),
     +         ec(ivmax,iregnmax),
     +         hwo(iregnmax),hwij(iregnmax),hwe(iregnmax)

           double precision Cimp
           integer
     +         iregn,mat
           integer
     +         NF
           character*16
     +         filein
```

```
      do 10 iregn=1,iregnmax

      if (iregn.eq.1) then
        mat = matc
        Cimp = ABS(par(2,(ipi(1)*3)/4))
      else if (iregn.eq.2) then
        mat = matb
        Cimp = ABS(par(2,(ipi(1)+ipi(2))/2))
      else if (iregn.eq.3) then
        mat = mate
        Cimp = ABS(par(2,ipi(2)+(M-ipi(2))/4))
      else
        STOP 'More device regions than expected'
      endif


      if (mat.le.1) then
        STOP 'Non available semiconductor material'
      else if (mat.eq.2) then
        filein='fparGaAs.dat'
      else if (mat.eq.3) then
        filein='fparInP.dat'
      else if (mat.eq.4) then
        filein='fparInGaAs.dat'
      else
        STOP 'Non available semiconductor material'
      endif

        call paramMCr(iregn,Cimp,tempK,mat,
     +          LNorm,parNorm,M,par,ivmax,iregnmax,ismmax,ienmax,
     +          de,iemax,efm,alf,gm,swk,ec,hwo,hwij,hwe,filein,NF)


10    continue


      return
      end

c************************ subroutine paramMCr ************************
c---------------------------------------------------------------------
c
c
c                   by Pau Garcias i Salva'. Nov., 1997
c
c---------------------------------------------------------------------
        subroutine paramMCr(iregn,Cimp,tempK,mat,
     +          LNorm,parNorm,M,par,ivmax,iregnmax,ismmax,ienmax,
     +          de,iemax,efm,alf,gm,swk,ec,hwo,hwij,hwe,filein,NF)
c
c       storage allocation
c
c             implicit double precision (a-h,o-z)
c             implicit integer (i-n)

        implicit none
c
            integer
     +          iregn,mat
        double precision
     +          Cimp,tempK
        integer
     +          LNorm,M,ivmax,iregnmax,ismmax,ienmax,iemax
        double precision
     +          parNorm(LNorm),par(19,0:M+1),de,efm(ivmax,iregnmax),
     +          alf(ivmax,iregnmax),gm(ivmax,iregnmax),
     +          swk(ivmax,ismmax,ienmax,iregnmax),
     +          ec(ivmax,iregnmax),
     +          hwo(iregnmax),hwij(iregnmax),hwe(iregnmax)

        double precision pi,q,hbar,bk,eps0,efm0
        double precision Cnorm,Dnorm,Vtnorm,tnorm,xnorm,
     +          vnorm,Enorm,Knorm,Fnorm,Alfnorm,
     +          He,Hk,Hv,Hke

        double precision
     +          ei,sei,ef,sef,qmax,qmin,ak,qq,wk,
     +          poe,poa,aco,
     +          ope,opa,
     +          eqe,eqa,
     +          qd2,bimp,
     +          rou,sv,z2,
     +          da,dij,deq,
     +          no,nij,ne,
     +          eps,epf,ep,
     +          eg,
     +          gmi,sriimax,AA
        integer
     +          i,ie,iv,iv2,ism1,ism2

        integer
     +          NF
        logical
     +          zexist
        character*16
     +          filein

    Cnorm   = parNorm(1)
    Dnorm   = parNorm(2)
    Vtnorm = parNorm(3)
    tnorm   = parNorm(4)
    xnorm   = parNorm(5)
    vnorm   = parNorm(6)
    Enorm   = parNorm(7)
    knorm   = parNorm(8)
    Fnorm   = parNorm(9)
    Alfnorm= parNorm(10)

    He    = parNorm(20)
    Hk    = parNorm(21)
    Hv    = parNorm(22)
    Hke   = parNorm(23)

    pi    = parNorm(30)
    q     = parNorm(31)
    hbar = parNorm(32)
    bk    = parNorm(33)
    eps0 = parNorm(34)
    efm0 = parNorm(35)
```

```
      OPEN (NF,STATUS='OLD',FORM='FORMATTED',FILE=filein,ERR=99)
      REWIND NF

        read(NF,*) efm(1,iregn)
        read(NF,*) efm(2,iregn)
        read(NF,*) alf(1,iregn)
        read(NF,*) alf(2,iregn)

        read(NF,*)

        read(NF,*) ec(1,iregn)
        read(NF,*) ec(2,iregn)
        read(NF,*) eg
        read(NF,*) eps
        read(NF,*) epf

        read(NF,*)

        read(NF,*) rou
        read(NF,*) sv
        read(NF,*) z2

        read(NF,*)

        read(NF,*) da
        read(NF,*) dij
        read(NF,*) deq

        read(NF,*)

        read(NF,*) hwo(iregn)
        read(NF,*) hwij(iregn)
        read(NF,*) hwe(iregn)

        read(NF,*)

        read(NF,*) sriimax

      close (NF)

      alf(1,iregn)=alf(1,iregn)/Alfnorm
      alf(2,iregn)=alf(2,iregn)/Alfnorm

      ec(1,iregn)=ec(1,iregn)/Enorm
      ec(2,iregn)=ec(2,iregn)/Enorm
      eg=eg/Enorm

      ep=1.0/(1.0/epf-1.0/eps)

      rou=rou*( (1.0e-02*xnorm)**3/efm0 )
      sv=sv/(1.0e-02*vnorm)

      da=da/Enorm
      dij=dij*1.0e-02*xnorm/Enorm
      deq=deq*1.0e-02*xnorm/Enorm

      hwo(iregn)=hwo(iregn)/Enorm
      hwij(iregn)=hwij(iregn)/Enorm
      hwe(iregn)=hwe(iregn)/Enorm

      sriimax=sriimax*tnorm

      no = 1.0 / (exp(hwo (iregn)) - 1.0)
      nij = 1.0 / (exp(hwij(iregn)) - 1.0)
      ne = 1.0 / (exp(hwe (iregn)) - 1.0)

      poe = q/(hbar*eps0*vnorm)*(no+1.)/(8.*pi)*
     +          hwo(iregn)/ep
      poa = poe*no/(no+1.)

      aco = (2.0d+04*pi*Enorm*Enorm)*
     +          ((q*tnorm)/(hbar*efm0*vnorm*vnorm))*
     +          da*da/(rou*sv*sv)

      ope = (1.0d+04*pi)*(q*hbar/efm0)*(tnorm/(xnorm*xnorm))*
     +          dij*dij/(hwij(iregn)*rou)*
     +          (nij+1.)
      opa = ope*nij/(nij+1.)

      eqe = (1.0d+04*pi)*(q*hbar/efm0)*(tnorm/(xnorm*xnorm))*
     +          deq*deq/(hwe(iregn)*rou)*
     +          (ne+1.)
      eqa = eqe*ne/(ne+1.)

      qd2  = (q/eps0/Enorm)*(Cnorm*xnorm*xnorm)*
     +          (Cimp/eps)
      bimp = (2.*pi/Enorm)*(q/eps0)*(q/eps0)/hbar*
     +          (Cnorm*xnorm*tnorm)*
     +          Cimp/(eps*eps)
      AA   = sqrt(2.*efm0*Enorm/q)*(1.0e-02*xnorm/hbar)


c-----------------------------------------------------------------------
c  BEGIN: Calculation of scattering rates - common mechanisms  ---------
c-----------------------------------------------------------------------

      do 20 ie=1,iemax

      ei=de*ie
      sei=sqrt(ei)

c  Gamma-valley
c  ------------

      iv=1
      iv2=2

c  Polar optical phonon - emission

        ef=ei-hwo(iregn)
        if (ef.gt.0.) then
          sef=sqrt(ef)
          qmax=sef+sei
          qmin=sei-sef
          swk(iv,1,ie,iregn)=
     +          poe*sqrt(efm(iv,iregn)/ei)*
     +          log(qmax/qmin)*AA
        else
          swk(iv,1,ie,iregn)=0.0
      endif
c  Polar optical phonon - absorption

      ef=ei+hwo(iregn)
      sef=sqrt(ef)
      qmax=sef+sei
      qmin=sef-sei
      swk(iv,2,ie,iregn)= swk(iv,1,ie,iregn)+
     +        poa*sqrt(efm(iv,iregn)/ei)*
     +        log(qmax/qmin)*AA

c  Non-polar optical phonon - emission / intervalley

      ef=ei-hwij(iregn)+
     +                ec(iv,iregn)-ec(iv2,iregn)
      if (ef.gt.0.) then
        sef=sqrt(
     +        ef*(1.+alf(iv2,iregn)*ef))
        swk(iv,3,ie,iregn)= swk(iv,2,ie,iregn)+
     +        z2*ope*sef*
     +        (1.+2.*alf(iv2,iregn)*ef)*
     +        (AA*sqrt(efm(iv2,iregn)))**3 /4./pi/pi
      else
        swk(iv,3,ie,iregn)= swk(iv,2,ie,iregn)
      endif

c  Non-polar optical phonon - absorption / intervalley

      ef=ei+hwij(iregn)+
     +                ec(iv,iregn)-ec(iv2,iregn)
      if (ef.gt.0.) then
        sef=sqrt(
     +        ef*(1.+alf(iv2,iregn)*ef))
        swk(iv,4,ie,iregn)= swk(iv,3,ie,iregn)+
     +        z2*opa*sef*
     +        (1.+2.*alf(iv2,iregn)*ef)*
     +        (AA*sqrt(efm(iv2,iregn)))**3 /4./pi/pi
      else
        swk(iv,4,ie,iregn)= swk(iv,3,ie,iregn)
      endif

c  Acoustic phonon

      ef=ei
      sef=sqrt(
     +        ef*(1.+alf(iv,iregn)*ef))
      swk(iv,5,ie,iregn)= swk(iv,4,ie,iregn)+
     +        aco*sef*
     +        (1.+2.*alf(iv,iregn)*ef)*
     +        (AA*sqrt(efm(iv,iregn)))**3 /4./pi/pi

c  Impurity scattering

      ef=ei
      sef=sqrt(
     +        ef*(1.+alf(iv,iregn)*ef))
      ak=AA*sef*sqrt(efm(iv,iregn))
      qq=qd2*(4.0*ak*ak+qd2)
      wk=bimp/qq*sef*
     +        (1.+2.*alf(iv,iregn)*ef)*
     +        (AA*sqrt(efm(iv,iregn)))**3 /4./pi/pi
      if (wk.gt.sriimax) wk=sriimax
      swk(iv,6,ie,iregn)=swk(iv,5,ie,iregn)+wk


c  L-valleys
c  ----------

      iv=2
      iv2=1

c  Polar optical phonon - emission

      ef=ei-hwo(iregn)
      if (ef.gt.0.) then
        sef=sqrt(ef)
        qmax=sef+sei
        qmin=sei-sef
        swk(iv,1,ie,iregn)=
     +        poe*sqrt(efm(iv,iregn)/ei)*
     +        log(qmax/qmin)*AA
      else
        swk(iv,1,ie,iregn)=0.0
      endif

c  Polar optical phonon - absorption

      ef=ei+hwo(iregn)
      sef=sqrt(ef)
      qmax=sef+sei
      qmin=sef-sei
      swk(iv,2,ie,iregn)= swk(iv,1,ie,iregn)+
     +        poa*sqrt(efm(iv,iregn)/ei)*
     +        log(qmax/qmin)*AA

c  Non-polar optical phonon - emission / equiv. valleys

      ef=ei-hwe(iregn)
      if (ef.gt.0.) then
        sef=sqrt(
     +        ef*(1.+alf(iv,iregn)*ef))
        swk(iv,3,ie,iregn)= swk(iv,2,ie,iregn)+
     +        (z2-1.)*eqe*sef*
     +        (1.+2.*alf(iv,iregn)*ef)*
     +        (AA*sqrt(efm(iv,iregn)))**3 /4./pi/pi
      else
        swk(iv,3,ie,iregn)= swk(iv,2,ie,iregn)
      endif

c  Non-polar optical phonon - absorption / equiv. valleys

      ef=ei+hwe(iregn)
      sef=sqrt(
     +        ef*(1.+alf(iv,iregn)*ef))
      swk(iv,4,ie,iregn)= swk(iv,3,ie,iregn)+
     +        (z2-1.)*eqa*sef*
     +        (1.+2.*alf(iv,iregn)*ef)*
     +        (AA*sqrt(efm(iv,iregn)))**3 /4./pi/pi

c  Non-polar optical phonon - emission / intervalley
```

```
        ef=ei-hwij(iregn)+
     +                ec(iv,iregn)-ec(iv2,iregn)
        if (ef.gt.0.) then
          sef=sqrt(
     +          ef*(1.+alf(iv2,iregn)*ef))
          swk(iv,5,ie,iregn)= swk(iv,4,ie,iregn)+
     +          ope*sef*
     +          (1.+2.*alf(iv2,iregn)*ef)*
     +          (AA*sqrt(efm(iv2,iregn)))**3 /4./pi/pi
        else
          swk(iv,5,ie,iregn)= swk(iv,4,ie,iregn)
        endif

c  Non-polar optical phonon - absorption / intervalley

        ef=ei+hwij(iregn)+
     +                ec(iv,iregn)-ec(iv2,iregn)
        if (ef.gt.0.) then
          sef=sqrt(
     +          ef*(1.+alf(iv2,iregn)*ef))
          swk(iv,6,ie,iregn)= swk(iv,5,ie,iregn)+
     +          opa*sef*
     +          (1.+2.*alf(iv2,iregn)*ef)*
     +          (AA*sqrt(efm(iv2,iregn)))**3 /4./pi/pi
        else
          swk(iv,6,ie,iregn)= swk(iv,5,ie,iregn)
        endif

c  Acoustic phonon

        ef=ei
        sef=sqrt(
     +          ef*(1.+alf(iv,iregn)*ef))
        swk(iv,7,ie,iregn)= swk(iv,6,ie,iregn)+
     +          aco*sef*
     +          (1.+2.*alf(iv,iregn)*ef)*
     +          (AA*sqrt(efm(iv,iregn)))**3 /4./pi/pi

c  Impurity scattering

        ef=ei
        sef=sqrt(
     +          ef*(1.+alf(iv,iregn)*ef))
        ak=AA*sef*sqrt(efm(iv,iregn))
        qq=qd2*(4.0*ak*ak+qd2)
        wk=bimp/qq*sef*
     +          (1.+2.*alf(iv,iregn)*ef)*
     +          (AA*sqrt(efm(iv,iregn)))**3 /4./pi/pi
        if (wk.gt.sriimax) wk=sriimax
        swk(iv,8,ie,iregn)=swk(iv,7,ie,iregn)+wk

20    continue

c------------------------------------------------------------------------
c  END: Calculation of scattering rates - common mechanisms  -----------
c------------------------------------------------------------------------

c------------------------------------------------------------------------
c  BEGIN: Calculation of scattering rates - specific mechanisms  -------
c------------------------------------------------------------------------

        ism1=6
        ism2=8

        if (mat .eq. 4) then
c        ism1=7
c        ism2=9
         print*, 'falta afegir alloy scattering'
c       else if (mat .eq. xx) then
c        ...

        endif

c------------------------------------------------------------------------
c  END: Calculation of scattering rates - specific mechanisms  ---------
c------------------------------------------------------------------------

        iv=1
        iv2=2

        gm(iv,iregn) =swk(iv,ism1,1,iregn)
        gm(iv2,iregn)=swk(iv2,ism2,1,iregn)

        do 30 ie=1,iemax

          if (swk(iv,ism1,ie,iregn) .gt. gm(iv,iregn))
     +                   gm(iv,iregn)=swk(iv,ism1,ie,iregn)

          if (swk(iv2,ism2,ie,iregn) .gt. gm(iv2,iregn))
     +                   gm(iv2,iregn)=swk(iv2,ism2,ie,iregn)

30      continue

        gmi = 1.0/gm(iv,iregn)

        do 40 i=1,ism1
          do 40 ie=1,iemax
             swk(iv,i,ie,iregn)=swk(iv,i,ie,iregn)*gmi
40      continue

        gmi= 1.0/gm(iv2,iregn)

        do 50 i=1,ism2
          do 50 ie=1,iemax
             swk(iv2,i,ie,iregn)=swk(iv2,i,ie,iregn)*gmi
50      continue


90    FORMAT (1X,3I5)
91    FORMAT (1X,3G23.16)


      return
99    INQUIRE (FILE= filein,EXIST=zexist)
      if (.not.zexist) then
         print*, filein,': File not found'
      else
         print*,'Error while opening file'
      endif
c
      stop 'Error trying to open/read input data file'
```

```
c      return
      end

c-----------------------------------------------------------------------
ceeee+i**#**** ****#**** ****#**** ****#**** ****#**** ****#**** ****#72------80
c-----------------------------------------------------------------------
```

```
      SUBROUTINE RCARIN(IJKL,RVEC,LENV)

C----------------------------------------------------------------------
C Inicializa valores antes de llamar a la subrutina RCARRY.
C IJKL debe estar en el rango 0<IJKL<900 000 000.
C Para conseguir los valores standar usados por Marsaglia y Zaman en su
C articulo poner IJKL = 54217137 (I=12, J=34, K=56, L=78)
C Version modificada (mas rapida que el original). (2/9/91)
c
c Canvis addicionals: Pau Garcias i Salva' (Dec. '97)
C----------------------------------------------------------------------
c      COMMON /RAN1/ CARRY
c      DIMENSION RVEC(LENV+24)

      implicit none

      integer
     +  IJKL,LENV
      real*4
     +  RVEC(0:LENV+24)

      integer
     +  IJ,KL,I,J,K,L,II,JJ,M
      real*4
     +  S,T,carry

      IJ = IJKL/30082
      KL = IJKL - 30082*IJ
      I = MOD(IJ/177,177) + 2
      J = MOD(IJ,177)     + 2
      K = MOD(KL/169,178) + 1
      L = MOD(KL,169)

      DO 2 II=24,1,-1
        S = 0.0
        T = 0.5
        DO 3 JJ=1,24
          M = MOD(MOD(I*J,179)*K,179)
          I = J
          J = K
          K = M
          L = MOD(53*L+1,169)
          IF (MOD(L*M,64).GE.32) S = S+T
          T = 0.5*T
3       CONTINUE
        RVEC(II) = S
2     CONTINUE

      CARRY = 0.0
      rvec(0) = carry

      RETURN
      END

      SUBROUTINE RCARRY(RVEC,LENV)
C----------------------------------------------------------------------
C Generador de numeros pseudo-aleatorios. Algoritmo de G. Marsaglia y
C A. Zaman. Genera numeros reales de 32-bits con mantisas de 24 bits,
C comprendidos entre 0 y 1 (1, explicitamente excluido).
C Periodo aproximado : 10**171.
C Admite la generacion de subsecuencias disjuntas.
C                  F. James, 1989
C                  Computer Physics Communications 60 (1990) 329-344
C Version modificada (mas rapida que el original). (2/9/91)
c
c Canvis addicionals: Pau Garcias i Salva' (Dec. '97)
C----------------------------------------------------------------------
c      DIMENSION RVEC(LENV+24)
c      COMMON /RAN1/ CARRY

      implicit none

      real*4
     +  TWOM24,twom48

      PARAMETER (TWOM24 = 1.0/16777216.0)
      parameter (twom48 = (2.**(-48)) )

      integer
     +  LENV
      real*4
     +  RVEC(0:LENV+24)

      integer
     +  IVEC,I
      real*4
     +  carry,UNI

C

      carry = rvec(0)

      DO 100 IVEC=25,LENV+24
        UNI = RVEC(IVEC-24) - RVEC(IVEC-10) - CARRY
        IF (UNI.LT.0.) THEN
          UNI = UNI + 1.0
          CARRY = TWOM24
        ELSE
          CARRY = 0.0
        ENDIF

        IF(UNI.EQ.0.)THEN
        UNI=RVEC(IVEC-24)*TWOM24
        IF(UNI.EQ.0.)UNI=twom48
        ENDIF

        RVEC(IVEC) = UNI
100   CONTINUE


      rvec(0) = carry

      DO 200 I=1,24
200   RVEC(I)=RVEC(LENV+I)

      RETURN
      END
```

```
ceeee+i**#**** ****#**** ****#**** ****#**** ****#**** ****#**** ****#72------80

c************************ subroutine initiaMC ************************
c---------------------------------------------------------------------
c
c
c                by Pau Garcias i Salva'. Nov., 1997
c
c---------------------------------------------------------------------
        subroutine initiaMC(nptmax,npt,pt,ipt,rn,epp,
     +          npcmax,npcmin,iwmax,ivmax,iregnmax,efm,alf,gm,
     +          LNorm,parNorm,M,par,nie,w,nbmax,ipi)
c
c       storage allocation
c
c            implicit double precision (a-h,o-z)
c            implicit integer (i-n)

        implicit none
c
c  parameters to limit the value of the initial energy of the new particles
c   to a reasonable value of 0.5eV
c    ei=-1.5*log(rmin+rmin1*rnd())
c
c    rnd()=]0.0 ... 1.0[,     rmin,    rmin1=1.0-rmin

            double precision
     +          rmin,rmin1,rmin2
            parameter(rmin=1.0d-20)
            parameter(rmin1=1.0d+00-rmin)

c  the same, in order to limit the free flight duration to a reasonable value
            parameter(rmin2=1.0d-50)

c
            integer
     +          base,base1,ntd
            parameter (base=10, base1=base-1, ntd=1)
c

            integer
     +          nptmax,npt,ipt(3,nptmax),ivmax,iregnmax,
     +          ipi(2),M,LNorm,nbmax,npcmax,npcmin,iwmax
            double precision
     +          pt(5,nptmax),epp,
     +          efm(ivmax,iregnmax),alf(ivmax,iregnmax),
     +          gm(ivmax,iregnmax),parNorm(LNorm),
     +          par(19,0:M+1),nie(2,M),w(nbmax,M)
            real*4
     +          rn(0:2+5+24,ntd)
            integer
     +          i,n,iv,iregn,i1,i2,iw,npc,n0,n1,n2
            double precision
     +          ei,ki,cs,sn,fai,
     +          Cnorm, xnorm,pi,He,epp1,
     +          efm1,alf1,gm1,rdx

        rdx=base
```

```
        Cnorm = parNorm(1)
        xnorm = parNorm(5)
        pi = parNorm(30)
        He = parNorm(20)

        epp1=(Cnorm*xnorm*1.0d+4)/epp
        npc=(npcmax+npcmin)/2

        npt=0

          par(1,0)=-par(1,2)
          par(18,0)=0.5*par(1,2)
          par(19,0)=-0.5*par(1,2)

          par(18,1)=par(1,2)
          par(19,1)=0.5*par(1,2)

        do 1 i=2,M-1
          par(18,i)=0.5*(par(1,i+1)-par(1,i-1))
          par(19,i)=0.5*(par(1,i)+par(1,i+1))
1       continue

          par(18,M)=(par(1,M)-par(1,M-1))
          par(19,M)=par(1,M)+0.5*(par(1,M)-par(1,M-1))

          par(1,M+1)=2.0*par(1,M)-par(1,M-1)
          par(18,M+1)=0.5*(par(1,M)-par(1,M-1))
          par(19,M+1)=2.0*par(1,M)-par(1,M-1)


        iv=1

        do 5 iregn=1,iregnmax

        efm1=efm(iv,iregn)
        alf1=alf(iv,iregn)
        gm1=-1./gm(iv,iregn)

c!!! cells 0 and M+1 will not be initialized

        if (iregn.eq.1) then
          i1=1
          i2=ipi(1)
        else if (iregn.eq.2) then
          i1=ipi(1)+1
          i2=ipi(2)
        else if (iregn.eq.3) then
          i1=ipi(2)+1
          i2=M
        else
          STOP 'more regions than expected in initiaMC()'
        endif


        do 10 i=i1,i2
          iw=iwmax
11        if ( (nie(1,i)*exp(w(1,i)-w(3,i))*par(18,i)*epp1)
     +                        .lt. (npcmax*(rdx**(iw-1))) ) then
              iw=iw-1
          goto 11
          endif
          n0=1+(nie(1,i)*exp(w(1,i)-w(3,i))*par(18,i)*epp1)*
     +          (rdx)**(-iw)

          if (n0 .ge. npc) then
               n2=npc+MOD(n0-npc,base)
               n1=(n0-npc)/base
               iw=iw+1
          else
               n1=n0-(1+(npc-n0)/base1)
               n2=base*(1+(npc-n0)/base1)
          endif


        do 20 n=npt+1, npt+n1
           call rcarry(rn(0,ntd),5)
           ei=-1.5*log(rmin+rmin1*rn(1,ntd))
           ki=sqrt(efm1*ei*(1.+alf1*ei)/He)

           cs=1.-2.*rn(2,ntd)
           sn=sqrt(1.-cs*cs)
           fai=2.*pi*rn(3,ntd)

           pt(1,n)=ki*cs
           pt(2,n)=ki*sn*cos(fai)
           pt(3,n)=ki*sn*sin(fai)
           pt(4,n)=gm1*log(rmin2+(1.0-rmin2)*rn(4,ntd))
           pt(5,n)=par(19,i)-par(18,i)*rn(5,ntd)
           ipt(1,n)=iv
           if (pt(5,n).le.par(1,i)) then
                ipt(2,n)=i
           else
                ipt(2,n)=i+1
           endif
           ipt(3,n)=iw
20      continue

          npt = npt + n1

        do 30 n=npt+1, npt+n2
           call rcarry(rn(0,ntd),5)
           ei=-1.5*log(rmin+rmin1*rn(1,ntd))
           ki=sqrt(efm1*ei*(1.+alf1*ei)/He)

           cs=1.-2.*rn(2,ntd)
           sn=sqrt(1.-cs*cs)
           fai=2.*pi*rn(3,ntd)

           pt(1,n)=ki*cs
           pt(2,n)=ki*sn*cos(fai)
           pt(3,n)=ki*sn*sin(fai)
           pt(4,n)=gm1*log(rmin2+(1.0-rmin2)*rn(4,ntd))
           pt(5,n)=par(19,i)-par(18,i)*rn(5,ntd)
           ipt(1,n)=iv
           if (pt(5,n).le.par(1,i)) then
                ipt(2,n)=i
           else
                ipt(2,n)=i+1
           endif
           ipt(3,n)=iw-1
30      continue

            npt = npt + n2

10        continue

5         continue

          if (npt.gt.nptmax) then
            write(*,*) 'The number of particles exceeds the allowed limit'
            write(*,*) '#particles=',npt,'> limit=',nptmax
            stop
          endif


        return
        end

c************************ subroutine MCdinam ************************
c-------------------------------------------------------------------
c
c
c                   by Pau Garcias i Salva'. Nov., 1997
c
c-------------------------------------------------------------------
          subroutine MCdinam(t,dt,npt,pt,ipt,rnc,rns,Lrns,Lbck,L2,nthds,
     +          LB,La,ivmax,iregnmax,ismmax,ienmax,iemax,nptmax,
     +          de1,efm,alf,gm,swk,ec,hwo,hwij,hwe,
     +          refp,E0,Teb,deT,ieTmax,matb,mate,jl,jh,ibt,
     +          nqj,nqjaux,nqjt,nqjtaux,znqj,wght,iwmax,nmax,
     +          nbmax,M,gradp,w,nie,par,parNorm,LNorm,ipi)
c
c         storage allocation
c
c             implicit double precision (a-h,o-z)
c             implicit integer (i-n)

          implicit none
c

c  parameter to limit the free flight duration to a reasonable value
              double precision
     +          rmin2
              parameter(rmin2=1.0d-50)


              integer
     +          ikx,iky,ikz,its,ix,ival,icel,iwgt
              parameter (ikx=1,iky=2,ikz=3,its=4,ix=5)
              parameter (ival=1,icel=2,iwgt=3)
c
              integer
     +          npt,ipt(3,nptmax),
     +          ivmax,iregnmax,ismmax,ienmax,iemax,nptmax,iwmax,nmax,
     +          LNorm,Lrns,Lbck,L2,nthds,La(L2,nthds),LB,
     +          M,ipi(2),nbmax,ieTmax,matb,mate,jl,jh(ivmax),ibt,
     +          nqj(nmax,ivmax),nqjaux(nmax,ivmax,nthds),
     +          nqjt(4,ivmax),nqjtaux(L2,ivmax,nthds)

              double precision
     +          t,dt,de1
              double precision
     +          pt(5,nptmax),wght(0:iwmax),
     +          efm(ivmax,iregnmax),alf(ivmax,iregnmax),
     +          gm(ivmax,iregnmax),
     +          swk(ivmax,ismmax,ienmax,iregnmax),
     +          ec(ivmax,iregnmax),
     +          hwo(iregnmax),hwij(iregnmax),hwe(iregnmax)
          real*4
     +          rnc(0:2+5+24,nthds),rns(0:3+Lrns*Lbck+24,nthds)
          double precision
     +          gradp(0:M+1),nie(2,M),par(19,0:M+1),
     +          refp(5),w(nbmax,M),
     +          E0(ivmax),Teb(ieTmax,ivmax),deT,
     +          parNorm(LNorm)

          integer
     +          n,ntd,k,kmax,iv,iw,j,jp,iregn
          double precision
     +          tdt,t1,tau,kx,ky,kz,ts,x

          logical
     +          zoutg,zout,znqj


       if (znqj) then
c$doacross local(k,iv,j),
c$& shared(nthds,ivmax,nqjtaux),
c$& mp_schedtype = simple
       do k=1,nthds
         do iv=1,ivmax
           do j=1,4
             nqjtaux(j,iv,k) = 0
           enddo
         enddo
       enddo

c$doacross local(k,iv,j),
c$& shared(nthds,ivmax,M,nqjaux),
c$& mp_schedtype = simple
       do k=1,nthds
         do iv=1,ivmax
           do j=1,M
             nqjaux(j,iv,k) = 0
           enddo
         enddo
       enddo
       endif

       tdt=t+dt

       kmax=npt/(nthds*LB)

c$ doacross local(ntd,k,n,kx,ky,kz,ts,x,jp,iv,iw,iregn,
c$& t1,tau,zout,zoutg),
c$& share(npt,La,Lbck,Lrns,rns,pt,ipt,ipi,t,tdt,gradp,efm,alf,gm,
c$& par,M,parNorm,LNorm,ivmax,iregnmax,rnc,del,swk,ec,hwo,hwij,hwe,
c$& w,nbmax,refp,E0,Teb,deT,ieTmax,matb,mate,jl,jh,ibt,
c$& nqjaux,nqjtaux,znqj,wght,iwmax,nmax,L2,
c$& ismmax,ienmax,iemax,nthds,kmax),
c$& mp_schedtype = simple
c$& ,affinity(ntd) = thread ( ntd-1 )
c..$& ,affinity(ntd) = data ( pt (1, 1+LB*(ntd-1+(k-1)*nthds)))
```

```
      do 110 ntd=1,nthds
      do 100 k=0,kmax
      do 20 n=1+LB*(ntd-1+k*nthds),
     +          min(npt, LB*(ntd+k*nthds))

      if (La(1,ntd).ge.Lbck) then
                La(1,ntd)=0
                call rcarry(rns(0,ntd),Lrns*Lbck)
      endif

        kx=pt(ikx,n)
        ky=pt(iky,n)
        kz=pt(ikz,n)
        ts=pt(its,n)
        x =pt(ix,n)
        jp   = ipt(icel,n)
        iv   = ipt(ival,n)
        iw   = ipt(iwgt,n)

        if (jp .le. ipi(1)) then
                iregn=1
        else if (jp .le. ipi(2)) then
                iregn=2
        else
                iregn=3
        endif

        t1=t

        zoutg=.false.
        zout=.false.

        if (ts.le.tdt) then

10         tau=ts-t1
           t1=ts

           call drift(tdt,tau,kx,ky,kz,ts,x,iv,jp,iregn,gradp,efm,alf,
     +          gm,ipi,par,M,parNorm,LNorm,ivmax,iregnmax,rnc,
     +          refp,E0,w,ec,Teb,deT,nbmax,ieTmax,matb,mate,jl,jh,ibt,
     +          nqjaux,nqjtaux,znqj,wght,iwmax,iw,nmax,L2,
     +          ntd,nthds,zout)
           if (zout) zoutg=.true.

c !!! n, t -->debugg
           call scatt(kx,ky,kz,iv,n,t,iregn,M,ipi,par,parNorm,
     +          del,efm,alf,gm,swk,ec,hwo,hwij,hwe,
     +          rns(La(1,ntd)*Lrns+1,ntd),
     +          ivmax,iregnmax,ismmax,ienmax,iemax,LNorm)

           ts=ts-log(rmin2+(1.0-rmin2)*rns(La(1,ntd)*Lrns+4,ntd))
     +           / gm(iv,iregn)

           La(1,ntd)=La(1,ntd)+1
           if (La(1,ntd).ge.Lbck) then
                La(1,ntd)=0
                call rcarry(rns(0,ntd),Lrns*Lbck)
           endif


        if (ts.le.tdt) goto 10
        endif

        tau=tdt-t1
        call drift(tdt,tau,kx,ky,kz,ts,x,iv,jp,iregn,gradp,efm,alf,
     +          gm,ipi,par,M,parNorm,LNorm,ivmax,iregnmax,rnc,
     +          refp,E0,w,ec,Teb,deT,nbmax,ieTmax,matb,mate,jl,jh,ibt,
     +          nqjaux,nqjtaux,znqj,wght,iwmax,iw,nmax,L2,
     +          ntd,nthds,zout)
        if (zout) zoutg=.true.



        pt(ikx,n)=kx
        pt(iky,n)=ky
        pt(ikz,n)=kz
        pt(its,n)=ts
        pt(ix,n) =x
        if (zoutg) then
                ipt(ival,n)=-iv
        else
                ipt(ival,n)=iv
        endif
        ipt(icel,n)=jp

20    continue
100   continue
110   continue

      if (znqj) then
      do k=1,nthds
        do iv=1,ivmax
          do j=1,4
            nqjt(j,iv) = nqjt(j,iv) + nqjtaux(j,iv,k)
          enddo
        enddo
      enddo

      do k=1,nthds
        do iv=1,ivmax
          do j=1,M
            nqj(j,iv) = nqj(j,iv) + nqjaux(j,iv,k)
          enddo
        enddo
      enddo
      endif

      return
      end
c************************ subroutine drift **************************
c--------------------------------------------------------------------
c
c
c                 by Pau Garcias i Salva'. Nov., 1997
c
c--------------------------------------------------------------------
       subroutine drift(tdt,tau,kx,ky,kz,ts,x,iv,jp,iregn,gradp,efm,alf,
     +          gm,ipi,par,M,parNorm,LNorm,ivmax,iregnmax,rn,
     +          refp,E0,w,ec,Teb,deT,nbmax,ieTmax,matb,mate,jl,jh,ibt,
     +          nqjaux,nqjtaux,znqj,wght,iwmax,iw,nmax,L2,
     +          ntd,nthds,zout)
c
c     storage allocation
c
c         implicit double precision (a-h,o-z)
c         implicit integer (i-n)

      implicit none
c
          integer
     +        ixm,ixup,iregnb,iregne
          parameter (ixm=1,ixup=19,iregnb=2,iregne=3)

c
          integer
     +        iv,jp,iregn,ipi(2),M,LNorm,ivmax,iregnmax,ntd,nthds,
     +        nbmax,ieTmax,matb,mate,jl,jh(ivmax),ibt,
     +        iwmax,iw,nmax,L2,
     +        nqjaux(nmax,ivmax,nthds),
     +        nqjtaux(L2,ivmax,nthds)

          double precision
     +        tdt,tau,kx,ky,kz,ts,x,deT,
     +        wght(0:iwmax)

          double precision
     +        gradp(0:M+1),
     +        refp(5),w(nbmax,M),ec(ivmax,iregnmax),
     +        Teb(ieTmax,ivmax),E0(ivmax),
     +        efm(ivmax,iregnmax),alf(ivmax,iregnmax),
     +        gm(ivmax,iregnmax),
     +        par(19,0:M+1),parNorm(LNorm)
          real*4
     +        rn(0:2+5+24,nthds)
          logical
     +        zout,znqj

          integer
     +        jpi,j
          double precision
     +        He,Hk,Hv,Hke,xj,xj1,force,
     +        Enorm,xi,kxi,ki2,kt2,kx2,
     +        exi,ex,exe,ek,eke,taue,taub


      Enorm  = parNorm(7)

      He     = parNorm(20)
      Hk     = parNorm(21)
      Hv     = parNorm(22)
      Hke    = parNorm(23)

      xi   = x
      kxi  = kx
      kt2  = ky*ky+kz*kz
      ki2  = kx*kx+kt2
      jpi  = jp

c  CIC scheme for force interpolation (Laux & Fischetti, 1991)
c
```

```
        xj = par(ixm,jp)
        xj1= par(ixm,jp-1)

        force = ( gradp(jp)*(x-xj1)+gradp(jp-1)*(xj-x) )/(xj-xj1)


c  Calculate the new position and momentum of the particle

        x=x+ Hv*tau* (kx+0.5*( Hk*force*tau )) /
     +    sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*
     +          ki2)*efm(iv,iregn))


        kx=kx+( Hk*force*tau )


c  Check for particle out of device boundaries:
c       - reflecting boundary conditions

        if (x.lt.par(ixm,0)) then
                zout=.true.
                x = 2.0*par(ixm,0)-x
                kx=ABS(kx)
                jp= 1
        else if (x.gt.par(ixup,M+1)) then
                zout=.true.
                x = 2.0*par(ixup,M+1)-x
                kx=-ABS(kx)
                jp= M+1
        endif


c  Spikes in HBTs

      if (matb.ne.mate) then
      if ((jpi.ge.jl).and.(jpi.le.jh(iv))) then
c## ELECTRON FROM EMITTER TO BASE (thermionic emission):
        if ((jpi.ge.ipi(2)+2).and.(x.lt.par(1,ipi(2)))) then
         exe= w(1,M) -
     +     ( w(1,jpi)*(xi-xj1)+
     +       w(1,jpi-1)*(xj-xi) )/(xj-xj1) +
     +     (sqrt(1.0+4.0*alf(iv,iregn)*He*kxi*kxi/efm(iv,iregn))-1.0)/
     +     (2.0*alf(iv,iregn))
         if (exe .gt. E0(iv)) then
          call rcarry(rn(0,ntd),1)
          if (rn(1,ntd).lt.Teb((exe-E0(iv))/deT+1,iv)) then
cc         transmission
           ek=
     +      (sqrt(1.0+4.0*alf(iv,iregne)*He*ki2/
     +       efm(iv,iregne))-1.0)/(2.0*alf(iv,iregne))-
     +       ( (-(w(1,ipi(2))+par(11,ipi(2)))+ec(iv,iregnb)) -
     +         (-((w(1,jpi)*(xi-xj1)+
     +             w(1,jpi-1)*(xj-xi) )/(xj-xj1) +
     +             par(11,jpi))+ec(iv,iregne)) )
           if (ek.gt.0.0) then
              kx2=((1.+2.*alf(iv,iregnb)*ek)**2-1.)*efm(iv,iregnb)/
     +          (4.*alf(iv,iregnb)*He) - kt2
              if (kx2 .gt. 0.0) then
                kx=-sqrt(kx2)
                taue=-((kxi/(Hk*force))+sqrt( (kxi/(Hk*force))**2 -
     +           2.0*(xi-par(1,ipi(2)))*
     +                 sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*ki2)*
     +                 efm(iv,iregn)) /(Hv*Hk*force) ))
                tau=tau-taue
                force=gradp(ipi(2))
                x=par(ixm,ipi(2))+ Hv*tau* (kx+0.5*( Hk*force*tau ))/
     +                 sqrt((efm(iv,iregnb)+4.0*alf(iv,iregnb)*He*
     +                 (kx*kx+kt2))*efm(iv,iregnb))
                kx=kx+( Hk*force*tau )
                if (x.gt.par(ixm,ipi(2))) x = 2.0*par(ixm,ipi(2))-x
                jp=ipi(2)
                if (znqj) nqjtaux(1,iv,ntd)=nqjtaux(1,iv,ntd)-wght(iw)
              else
cc             goto 1: reflection
                taue=-((kxi/(Hk*force))+sqrt( (kxi/(Hk*force))**2 -
     +           2.0*(xi-par(1,ipi(2)))*
     +           sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*ki2)*
     +           efm(iv,iregn)) /(Hv*Hk*force) ))
                kx= -( kxi+( Hk*force*taue ) )
c=              force = force
                tau=tau-taue
                x=par(ixm,ipi(2)+1)+ Hv*tau* (kx+0.5*( Hk*force*tau ))/
     +           sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*
     +           (kx*kx+kt2))*efm(iv,iregn))
                kx=kx+( Hk*force* tau )
                jp=ipi(2)+2
              endif
           else
cc          goto 1: reflection
             taue=-((kxi/(Hk*force))+sqrt( (kxi/(Hk*force))**2 -
     +          2.0*(xi-par(1,ipi(2)))*
     +          sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*ki2)*
     +          efm(iv,iregn)) /(Hv*Hk*force) ))
              kx= -( kxi+( Hk*force*taue ) )
c=            force = force
              tau=tau-taue
              x=par(ixm,ipi(2)+1)+ Hv*tau* (kx+0.5*( Hk*force*tau ))/
     +         sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*
     +         (kx*kx+kt2))*efm(iv,iregn))
              kx=kx+( Hk*force* tau )
              jp=ipi(2)+2
           endif
          else
cc         goto 1: reflection
            taue=-((kxi/(Hk*force))+sqrt( (kxi/(Hk*force))**2 -
     +          2.0*(xi-par(1,ipi(2)))*
     +          sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*ki2)*
     +          efm(iv,iregn)) /(Hv*Hk*force) ))
            kx= -( kxi+( Hk*force*taue ) )
c=          force = force
            tau=tau-taue
            x=par(ixm,ipi(2)+1)+ Hv*tau* (kx+0.5*( Hk*force*tau ))/
     +       sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*
     +       (kx*kx+kt2))*efm(iv,iregn))
            kx=kx+( Hk*force* tau )
            jp=ipi(2)+2
          endif
         else
cc        reflection
1          taue=-((kxi/(Hk*force))+sqrt( (kxi/(Hk*force))**2 -
     +          2.0*(xi-par(1,ipi(2)))*
     +          sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*ki2)*
     +          efm(iv,iregn)) /(Hv*Hk*force) ))
           kx= -( kxi+( Hk*force*taue ) )
c=         force = force
           tau=tau-taue
           x=par(ixm,ipi(2)+1)+ Hv*tau* (kx+0.5*( Hk*force*tau ))/
     +      sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*
     +      (kx*kx+kt2))*efm(iv,iregn))
           kx=kx+( Hk*force* tau )
           jp=ipi(2)+2
         endif
c## ELECTRON FROM BASE TO EMITTER (thermionic or tunneling transm.):
        else if ((jpi.le.ipi(2)).and.(x.gt.par(1,ipi(2)))) then
         eke=
     +      (sqrt(1.0+4.0*alf(iv,iregnb)*He*ki2/
     +       efm(iv,iregnb))-1.0)/(2.0*alf(iv,iregnb))-
     +       ( (-(w(1,M)+par(11,M))+ec(iv,iregne)) -
     +         (-((w(1,jpi)*(xi-xj1)+
     +             w(1,jpi-1)*(xj-xi) )/(xj-xj1) +
     +             par(11,jpi))+ec(iv,iregnb)) )
         taub=-(kxi/(Hk*force))+sqrt( (kxi/(Hk*force))**2 -
     +      2.0*(xi-par(1,ipi(2)))*
     +      sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*ki2)*
     +      efm(iv,iregn)) /(Hv*Hk*force) )
         if ( eke.gt.0.0 ) then
          kx2=((1.+2.*alf(iv,iregne)*eke)**2-1.)*efm(iv,iregne)/
     +          (4.*alf(iv,iregne)*He) - kt2
          if (kx2 .gt. 0.) then
           exe=
     +      (sqrt(1.0+4.0*alf(iv,iregne)*He*kx2/
     +       efm(iv,iregne))-1.0)/(2.0*alf(iv,iregne))
           if ( exe.gt.E0(iv) ) then
            call rcarry(rn(0,ntd),1)
            if (rn(1,ntd).lt.Teb((exe-E0(iv))/deT+1,iv)) then
cc              transmission

                jp=ipi(2)+1
                ek=eke - (-w(1,jp)+w(1,M))

                if (ek.lt.0.0) then
5                jp=jp+1
                 ek=ek+(w(1,jp)-w(1,jp-1))
                 if (ek.lt.0.0) goto 5
                endif

                kx2=((1.+2.*alf(iv,iregne)*ek)**2-1.)*efm(iv,iregne)/
     +              (4.*alf(iv,iregne)*He) - kt2

                if (kx2.lt.0.0) then
6                jp=jp+1
                 ek=ek+(w(1,jp)-w(1,jp-1))
                 if (ek.lt.0.0) goto 6
                 kx2=((1.+2.*alf(iv,iregne)*ek)**2-1.)*efm(iv,iregne)/
     +               (4.*alf(iv,iregne)*He) - kt2
                 if (kx2.lt.0.0) goto 6
                endif

                kx=sqrt(kx2)
                tau=tau-taub
                force=gradp(jp)
                x=par(ixm,jp)+ Hv*tau* (kx+0.5*( Hk*force*tau )) /
     +           sqrt((efm(iv,iregne)+4.0*alf(iv,iregne)*He*
     +            (kx*kx+kt2))*efm(iv,iregne))
                kx=kx+( Hk*force*tau )
                jp=jp+1
                if (znqj) then
                 if ( exe .gt. (-w(1,ipi(2)+1)+w(1,M)) ) then
                  nqjtaux(2,iv,ntd) = nqjtaux(2,iv,ntd) + wght(iw)
                 else
                  nqjtaux(4,iv,ntd) = nqjtaux(4,iv,ntd) + wght(iw)
                 endif
                endif
            else
cc           goto 2: reflection .- rn().ge.Teb()
              kx= -( kxi+( Hk*force*taub ) )
c=            force = force
              tau=tau-taub
              x=par(ixm,ipi(2))+ Hv*tau* (kx+0.5*( Hk*force*tau ))/
     +         sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*
     +         (kx*kx+kt2))*efm(iv,iregn))
              kx=kx+( Hk*force* tau )
              if (x.gt.par(ixm,ipi(2))) x = 2.0*par(ixm,ipi(2))-x
              jp=ipi(2)
            endif
           else
cc          goto 2: reflection .- exe .le. E0(iv)
             kx= -( kxi+( Hk*force*taub ) )
c=           force = force
             tau=tau-taub
             x=par(ixm,ipi(2))+ Hv*tau* (kx+0.5*( Hk*force*tau ))/
     +        sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*
     +        (kx*kx+kt2))*efm(iv,iregn))
             kx=kx+( Hk*force* tau )
             if (x.gt.par(ixm,ipi(2))) x = 2.0*par(ixm,ipi(2))-x
             jp=ipi(2)
           endif
          else
cc         goto 2: reflection .- kx2.le.0.0
            kx= -( kxi+( Hk*force*taub ) )
c=          force = force
            tau=tau-taub
            x=par(ixm,ipi(2))+ Hv*tau* (kx+0.5*( Hk*force*tau ))/
     +       sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*
     +       (kx*kx+kt2))*efm(iv,iregn))
            kx=kx+( Hk*force* tau )
            if (x.gt.par(ixm,ipi(2))) x = 2.0*par(ixm,ipi(2))-x
            jp=ipi(2)
          endif
         else
cc        reflection .- eke .le. 0.0
2          kx= -( kxi+( Hk*force*taub ) )
c=         force = force
           tau=tau-taub
           x=par(ixm,ipi(2))+ Hv*tau* (kx+0.5*( Hk*force*tau ))/
     +      sqrt((efm(iv,iregn)+4.0*alf(iv,iregn)*He*
     +      (kx*kx+kt2))*efm(iv,iregn))
           kx=kx+( Hk*force* tau )
           if (x.gt.par(ixm,ipi(2))) x = 2.0*par(ixm,ipi(2))-x
           jp=ipi(2)
         endif
c## ELECTRON FROM EMITTER TO BASE (tunneling transm.):
        else if ((kxi*kx).le.0.0) then
         if (xi.ge.par(1,ipi(2))) then
```

```
           exe= w(1,M) -
     +      ( w(1,jpi)*(xi-xj1)+
     +        w(1,jpi-1)*(xj-xi) )/(xj-xj1) +
     +      (sqrt(1.0+4.0*alf(iv,iregn)*He*kxi*kxi/efm(iv,iregn))-1.0)/
     +      (2.0*alf(iv,iregn))
           if ( exe.gt.E0(iv) ) then
            call rcarry(rn(0,ntd),1)
            if (rn(1,ntd).lt.Teb((exe-E0(iv))/deT+1,iv)) then
cc               tunneling transmission
             ek=
     +         (sqrt(1.0+4.0*alf(iv,iregne)*He*ki2/
     +          efm(iv,iregne))-1.0)/(2.0*alf(iv,iregne))-
     +          ( (-(w(1,ipi(2))+par(11,ipi(2)))+ec(iv,iregnb)) -
     +            (-((w(1,jpi)*(xi-xj1)+
     +                w(1,jpi-1)*(xj-xi) )/(xj-xj1) +
     +                par(11,jpi))+ec(iv,iregne)) )
             if (ek.gt.0.0) then
               kx2=((1.+2.*alf(iv,iregnb)*ek)**2-1.)*efm(iv,iregnb)/
     +            (4.*alf(iv,iregnb)*He) - kt2
               if (kx2 .gt. 0.0) then
                  kx=-sqrt(kx2)
                  tau=tau+kxi/(Hk*force)
                  force=gradp(ipi(2))
                  x=par(ixm,ipi(2))+ Hv*tau* (kx+0.5*( Hk*force*tau )) /
     +             sqrt((efm(iv,iregnb)+4.0*alf(iv,iregnb)*He*
     +             (kx*kx+kt2))*efm(iv,iregnb))
                  kx=kx+( Hk*force*tau )
                  if (x.gt.par(ixm,ipi(2))) x = 2.0*par(ixm,ipi(2))-x
                  jp=ipi(2)
                  if (znqj) nqjtaux(3,iv,ntd)=nqjtaux(3,iv,ntd)-wght(iw)
               endif
             endif
c           else
cc               accept the reflection
c                continue
            endif
           endif
c         else
c          continue
          endif
         endif
        endif
        endif

c  continue with the normal case

         if (x.gt.par(ixm,jp)) then
10               jp= jp+1
                 if (x.gt.par(ixm,jp)) goto 10
         else if (x.lt.par(ixm,jp-1)) then
20               jp= jp-1
                 if (x.lt.par(ixm,jp-1)) goto 20
         endif

         if (znqj) then
           if (jp .gt. jpi) then
             do j=jpi,jp-1
               nqjaux(j,iv,ntd) = nqjaux(j,iv,ntd) + wght(iw)
             enddo
           else if (jp .lt. jpi) then
             do j=jp,jpi-1
               nqjaux(j,iv,ntd) = nqjaux(j,iv,ntd) - wght(iw)
             enddo
           endif
         endif

         if ( jp .gt. ( ipi(2)+1 ) ) then
               iregn=3
         else if ( jp .le. ipi(1) ) then
               iregn=1
         else
               iregn=2
         endif


       return
       end

c*************************  subroutine tunnel  ***********************
c--------------------------------------------------------------------
c
c          FORTRAN subroutine to calculate the transmission coeff.
c            through an energy spike in the conduction band
c
c                            Jan., 1999
c
c--------------------------------------------------------------------
         subroutine tunnel (Teb,deT,w,par,ec,efm,alf,E0,refp,Alfnorm,
     +           Enorm,xnorm,M,ipi,ibt,nbmax,ivmax,iregnmax,ieTmax,jh)
c
c        storage allocation
c
c             implicit double precision (a-h,o-z)
c             implicit integer (i-n)
c
         implicit none

             integer
     +           ixm,idx,iregnb,iregne
             parameter (ixm=1,idx=18,iregnb=2,iregne=3)

c Ep! Sistema d'unitats MKS
             double precision pi,q,hbar,bk,eps0,efm0,hbar2
             parameter (pi   = 3.14159265d+00, q    = 1.6021892d-19)
             parameter (hbar = 1.05459d-34,    bk   = 1.38066d-23)
             parameter (eps0 = 8.85419d-12,    efm0 = 9.10953d-31)
             parameter (hbar2= 1.11216d-68)

             double precision Eyze
             parameter (Eyze=26.d-03 * q)

             integer
     +           M,ipi(2),ibt,nbmax,ivmax,iregnmax,ieTmax,jh(ivmax)

             double precision
     +           w(nbmax,M),par(19,0:M+1),refp(5),
     +           deT,Alfnorm,Enorm,xnorm,
     +           ec(ivmax,iregnmax),efm(ivmax,iregnmax),
     +           alf(ivmax,iregnmax),
     +           Teb(ieTmax,ivmax),E0(ivmax)
```

```
        integer
     +        i,j,iv

        double precision
     +        efme,efmb,alfe,alfb,
     +        Ee,Eb,Ex,Eke,Er,Eyzb,E0q,qdeT,uj,dl0,kt2,k02,
     +        s11,s22,s12,s21,t11,t22,t12,t21,
     +        dl,bt,af,Sh,Ch,sn,cs,k0,kn


      qdeT=q*deT*Enorm
      dl0=xnorm*1.0d-02


c   Upper boundary for electrons near the heterojunction

      do iv=1,ivmax

       efme=efm(iv,iregne)*efm0
       efmb=efm(iv,iregnb)*efm0
       alfe=alf(iv,iregne)*Alfnorm/q
       alfb=alf(iv,iregnb)*Alfnorm/q

       Ee=q*(-refp(4)+Enorm*(-(w(1,M)+par(11,M))+ec(iv,iregne)))
       Eb=q*(-refp(4)+
     +     Enorm*(-(w(1,ipi(2))+par(11,ipi(2)))+ec(iv,iregnb)))
       uj=q*(-refp(4)+
     +     Enorm*(-(w(1,ipi(2)+1)+par(11,ipi(2)+1))+ec(iv,iregne)))

       kt2 = (Eyze*(1.+alfe*Eyze)*2.*efme) / hbar2
       Eyzb=(sqrt(1.+2.*alfb*hbar2*kt2/efmb)-1.)/(2.*alfb)

       E0q= qdeT
       if ((Eb+Eyzb-Ee) .gt.0.0) then
         k02=(((1.+2.*alfe*(Eb+Eyzb-Ee))**2-1.)*efme)/
     +        (2.*alfe*hbar2) - kt2
         if (k02 .gt. 0.0) then
           E0q= qdeT+(sqrt(1.+2.*alfe*hbar2*k02/efme)-1.)/(2.*alfe)
         endif
       endif

       E0(iv) = E0q/(q*Enorm)

       Er = Ee + E0q
       Er = max ( Er , Ee + 0.05*(uj-Ee))
       Er = min ( Er , uj - 0.05*(uj-Ee))

        uj=q*(-refp(4)+
     +     Enorm*(-(w(1,jh(iv))+par(11,jh(iv)))+ec(iv,iregne)))

      if (uj .gt. Er) then
5       jh(iv) =jh(iv)+1
        uj=q*(-refp(4)+
     +     Enorm*(-(w(1,jh(iv))+par(11,jh(iv)))+ec(iv,iregne)))
        if (uj .gt. Er) goto 5
      else if (uj .lt. Er) then
6       jh(iv) =jh(iv)-1
        uj=q*(-refp(4)+
     +     Enorm*(-(w(1,jh(iv))+par(11,jh(iv)))+ec(iv,iregne)))
        if (uj .lt. Er) goto 6
      endif
```

```
        j=1
7       if ( ((jh(iv)+j) .lt. (M-1)) .and.
     +   ( par(ixm,jh(iv)+j) .lt. par(ixm,jh(iv))+100 ) ) then
         j=j+1
         goto 7
        endif

        jh(iv)=jh(iv)+j


c   Compute the transmission coefficient

         do i = 1, ieTmax

            Ex = E0q + (i-1)*qdeT

            s11=1.0
            s12=0.0
            s21=0.0
            s22=1.0


            do j=ipi(2)+1, M
             uj=-refp(4)+Enorm*(-(w(1,j)+par(11,j))+ec(iv,iregne))
             uj=q*uj
             if ( uj .gt. Ex+Ee ) then
              bt = sqrt((2.*efme/hbar2)*(uj-(Ex+Ee))*
     +                    (1.+alfe*(uj-(Ex+Ee))))
              dl = dl0*par(idx,j)
              Sh = 0.5*(exp(bt*dl)-exp(-bt*dl))
              Ch = 0.5*(exp(bt*dl)+exp(-bt*dl))
              t11=s11
              t12=s12
              t21=s21
              t22=s22

              s11 = t11 * Ch + t12 * bt/efme * Sh
              s12 = t11 * efme/bt * Sh + t12 * Ch
              s21 = t21 * Ch + t22 * bt/efme * Sh
              s22 = t21 * efme/bt * Sh + t22 * Ch

             else if ( uj .lt. Ex+Ee ) then
              af = sqrt((2.*efme/hbar2)*(-uj+(Ex+Ee))*
     +                    (1.+alfe*(-uj+(Ex+Ee))))
              dl = dl0*par(idx,j)
              sn = sin(af*dl)
              cs = cos(af*dl)
              t11=s11
              t12=s12
              t21=s21
              t22=s22

              s11 = t11 * cs - t12 * af/efme * sn
              s12 = t11 * efme/af * sn + t12 * cs
              s21 = t21 * cs - t22 * af/efme * sn
              s22 = t21 * efme/af * sn + t22 * cs
             endif
c                I assume sij= Identity matrix when Ex-Etg = uj
```

```
            enddo

            k0 = sqrt((2.*efme/hbar2)*Ex*(1.+alfe*Ex))
            Eke=(sqrt(1.+2.*alfe*hbar2/efme*(k0*k0+kt2))-1.)/
     +        (2.*alfe)
            kn=sqrt((((1.+2.*alfb*(Ee+Eke-Eb))**2-1.)*efmb)/
     +        (2.*alfb*hbar2) - kt2)

            Teb(i,iv) = 4.0*efmb*k0/(efme*kn)/
     +      ((s11+s22*efmb*k0/(efme*kn))**2 +
     +       (s12*k0/efme-s21*efmb/kn)**2)

         enddo
        enddo

      return
      end

c*********************** subroutine scatt **************************
c------------------------------------------------------------------
c
c
c                 by Pau Garcias i Salva'. Nov., 1997
c
c------------------------------------------------------------------
        subroutine scatt(kx,ky,kz,iv,n,t,iregn,M,ipi,par,parNorm,
     +          del,efm,alf,gm,swk,ec,hwo,hwij,hwe,rn,
     +          ivmax,iregnmax,ismmax,ienmax,iemax,LNorm)
c
c       storage allocation
c
c          implicit double precision (a-h,o-z)
c          implicit integer (i-n)

        implicit none
c
           integer
     +          iv,n,iregn,M,ipi(2),
     +          ivmax,iregnmax,ismmax,ienmax,iemax,LNorm

            double precision
     +          kx,ky,kz,del,t,
     +          efm(ivmax,iregnmax),alf(ivmax,iregnmax),
     +          gm(ivmax,iregnmax),
     +          swk(ivmax,ismmax,ienmax,iregnmax),
     +          ec(ivmax,iregnmax),
     +          hwo(iregnmax),hwij(iregnmax),hwe(iregnmax),
     +          par(19,0:M+1),parNorm(LNorm)
           real*4
     +          rn(3)

           integer
     +          ie,iv2
            double precision
     +          pi,q,eps0,Enorm,Cnorm,xnorm,Hke,He,
     +          ki2,ki,ei,kf,ef,qd2i,r1,r2,f,cb,sb,fai,skk,
     +          a11,a12,a13,a21,a22,a23,a32,a33,x1,x2,x3,
     +          efm1,alf2

        efm1 = 1.0 /efm(iv,iregn)
```

```
        alf2 = 1.0 /(2.*alf(iv,iregn))

        Cnorm  = parNorm(1)
        xnorm  = parNorm(5)
        Enorm  = parNorm(7)

        He    = parNorm(20)
        Hke   = parNorm(23)

        pi    = parNorm(30)
        q     = parNorm(31)
        eps0  = parNorm(34)


        ki2=(kx*kx+ky*ky+kz*kz)

        ei=(sqrt(1.+4.*alf(iv,iregn)*He*
     +          ki2 * efm1 ) -1.0) * alf2


        ie=NINT(ei*del)

        if (ie.le.0) return

        if (ie.ge.iemax) then
          write(*,*)
     +  'Electron energy higher than the maximum allowed value, iemax'
          write(*,*)
     +  'Energy has been reduced to 90%*iemax to proceed the simulation'
          write(*,*) ie, iemax

          write(*,*) kx,ky,kz,iv,n,t
          STOP

          ie=0.9*iemax
          ei=ie/del
          ki2=(efm(iv,iregn)*ei*(1.0+alf(iv,iregn)*ei)) / He
        endif


        ki=sqrt(ki2)

c   qd2i = 1.0 / qd2, used only for computational efficiency
        if (iregn.eq.3) then
                qd2i  = (par(15,ipi(2)+(M-ipi(2))/4)*eps0*Enorm) /
     +                  (q*ABS(par(2,ipi(2)+(M-ipi(2))/4))*
     +                  (Cnorm*xnorm*xnorm))
c               goto 3000
        else if (iregn.eq.1) then
                qd2i  = (par(15,(ipi(1)*3)/4)*eps0*Enorm) /
     +                  (q*ABS(par(2,(ipi(1)*3)/4))*
     +                  (Cnorm*xnorm*xnorm))
c               goto 1000
             else
                qd2i  = (par(15,(ipi(1)+ipi(2))/2)*eps0*Enorm) /
     +                  (q*ABS(par(2,(ipi(1)+ipi(2))/2))*
     +                  (Cnorm*xnorm*xnorm))
c               goto 2000
c            endif
        endif
```

```
c  particle in the collector region of the device
c1000   if (iv.eq.1) then


c  particle in the base region of the device
c2000   if (iv.eq.1) then


c  particle in the emitter region of the device
c3000   if (iv.eq.1) then


        if (iv.eq.1) then
          iv2  =2

          r1=rn(1)

          if (r1.gt.swk(iv,6,ie,iregn)) return

          if (r1.le.swk(iv,1,ie,iregn)) then
                ef=ei-hwo(iregn)
                if (ef.le.0.) return
                goto 20
          else if (r1.le.swk(iv,2,ie,iregn)) then
                ef=ei+hwo(iregn)
                goto 20
          else if (r1.le.swk(iv,3,ie,iregn)) then
                ef=ei-hwij(iregn)+ec(iv,iregn)-ec(iv2,iregn)
                if (ef.le.0.) return
                iv=iv2
                goto 40
          else  if (r1.le.swk(iv,4,ie,iregn)) then
                ef=ei+hwij(iregn)+ec(iv,iregn)-ec(iv2,iregn)
                if (ef.le.0.) return
                iv=iv2
                goto 40
          else  if (r1.le.swk(iv,5,ie,iregn)) then
                ef=ei
cc              kf=ki
                goto 40
          else  if (r1.le.swk(iv,6,ie,iregn)) then
                ef=ei
                r2=rn(2)
                cb=1.-r2/(0.5+(1.-r2)*2.*ki*ki*qd2i)
                kf=ki
                goto 30
          endif
c       endif
c
c else  if (iv.eq.2) then

        else
          iv2  =1

          r1=rn(1)

          if (r1.gt.swk(iv,8,ie,iregn)) return

          if (r1.le.swk(iv,1,ie,iregn)) then
                ef=ei-hwo(iregn)
                if (ef.le.0.) return
                goto 20
          else if (r1.le.swk(iv,2,ie,iregn)) then
                ef=ei+hwo(iregn)
                goto 20
          else if (r1.le.swk(iv,3,ie,iregn)) then
                ef=ei-hwe(iregn)
                if (ef.le.0.) return
                goto 40
          else if (r1.le.swk(iv,4,ie,iregn)) then
                ef=ei+hwe(iregn)
                goto 40
          else if (r1.le.swk(iv,5,ie,iregn)) then
                ef=ei-hwij(iregn)+ec(iv,iregn)-ec(iv2,iregn)
                if (ef.le.0.) return
                iv=iv2
                goto 40
          else  if (r1.le.swk(iv,6,ie,iregn)) then
                ef=ei+hwij(iregn)+ec(iv,iregn)-ec(iv2,iregn)
                if (ef.le.0.) return
                iv=iv2
                goto 40
          else  if (r1.le.swk(iv,7,ie,iregn)) then
                ef=ei
cc              kf=ki
                goto 40
          else  if (r1.le.swk(iv,8,ie,iregn)) then
                ef=ei
                r2=rn(2)
                cb=1.-r2/(0.5+(1.-r2)*2.*ki*ki*qd2i)
                kf=ki
                goto 30
          endif
        endif

c       stop 'Non expected valley-index in subroutine scatt()'
        write (*,*) 'Non expected valley-index in subroutine scatt()'
        write (*,*) n,iv,iregn,t
        write (*,*)

        STOP



c   Determination of final states

20      kf=Hke*sqrt(efm(iv,iregn)*ef*(1.0+alf(iv,iregn)*ef))
        f =2.0*ki*kf/((ki-kf)*(ki-kf))
        if (f.le.0.) return
        cb=(1.0+f-(1.0+2.0*f)**rn(2))/f

30      sb=sqrt(1.0-cb*cb)
        fai=2.0*pi*rn(3)
        skk=sqrt(kx*kx+ky*ky)

        a11=ky/skk
        a12=kx*kz/(skk*ki)
        a13=kx/ki

        a21=-kx/skk
        a22=ky*kz/(skk*ki)
```

```
      a23=ky/ki

      a32=-skk/ki
      a33=kz/ki

      x1=kf*sb*cos(fai)
      x2=kf*sb*sin(fai)
      x3=kf*cb

      kx=a11*x1+a12*x2+a13*x3
      ky=a21*x1+a22*x2+a23*x3
      kz=       a32*x2+a33*x3

      return

40    kf=Hke*sqrt(efm(iv,iregn)*ef*(1.0+alf(iv,iregn)*ef))
      cb=1.0-2.0*rn(2)
      sb=sqrt(1.0-cb*cb)
      fai=2.0*pi*rn(3)

      kx=kf*cb
      ky=kf*sb*cos(fai)
      kz=kf*sb*sin(fai)

      return

    end

c************************ subroutine chargeCIC ***********************
c--------------------------------------------------------------------
c
c
c                 by Pau Garcias i Salva'. Nov., 1997
c
c--------------------------------------------------------------------
      subroutine chargeCIC(pt,ipt,npt,nptmax,tdt,epp,cn0c,cn0e,cn,
     +        npti,nptiw,nptnw,ifree,ipi,w,par,M,efm,alf,gm,rn,
     +        cnaux,LB,nptcon,nptcon4,joutc,joute,joutcaux,jouteaux,
     +        nptiwaux,nptnwaux,jelim,wght,
     +        nmax,iwmax,inmax,lfrmax,lfrtop,npttop,nptitop,nthds,
     +        parNorm,LNorm,nbmax,ivmax,iregnmax,npcmax,npcmin)
c
c     storage allocation
c
c          implicit double precision (a-h,o-z)
c          implicit integer (i-n)

      implicit none
c
          integer
     +        ikx,iky,ikz,its,ix,ival,icel,iwgt,ixm,idx,ixup
          parameter (ikx=1,iky=2,ikz=3,its=4,ix=5)
          parameter (ival=1,icel=2,iwgt=3,ixm=1,idx=18,ixup=19)
c
          integer
     +        base,base1
          parameter (base=10, base1=base-1)
          double precision
     +        basei
          parameter (basei=1.0/base)
```

```
c
            integer
     +        M,nthds,LB,ipt(3,nptmax),npti(0:M+1),npt,nptmax,LNorm,
     +        nptiw(0:iwmax,0:M+1),inmax,nptnw(inmax,0:iwmax,0:M+1),
     +        nbmax,ivmax,iregnmax,nmax,iwmax,npcmax,npcmin,
     +        nptiwaux(0:iwmax,0:nmax+1,nthds),
     +        nptnwaux(inmax,0:iwmax,0:nmax+1,nthds),
     +        nptcon,nptcon4,
     +        jelim(nptcon4),joutc(nptcon),joute(nptcon),
     +        joutcaux(0:nptcon,nthds),
     +        jouteaux(0:nptcon,nthds),
     +        lfrmax,ifree(base1,lfrmax),ipi(2),
     +        lfrtop,npttop,nptitop

            double precision
     +        pt(5,nptmax),tdt,epp,cn0c,cn0e,
     +        cn(0:M+1),cnaux(0:nmax+1,nthds),wght(0:iwmax),
     +        efm(ivmax,iregnmax),alf(ivmax,iregnmax),
     +        gm(ivmax,iregnmax),
     +        par(19,0:M+1),parNorm(LNorm),w(nbmax,M)
            real*4
     +        rn(0:2+5+24,nthds)


            integer
     +        n,i,j,k,jp,nnptc,iv,iregn,iw,niw,lfree,iwtop,i1,i2,
     +        ntd,incr,increm,kmax,joc,joe,jocd,joed,jel
            double precision
     +        kx,ky,kz,ts,x,Cnorm,xnorm,epp1,
     +        xM,xM1,xM2,x2,x1,x0,xj,xj1,cnj,
     +        invfer

        external invfer

        increm=(npcmax-npcmin)/4

        Cnorm = parNorm(1)
        xnorm = parNorm(5)

        epp1=(Cnorm*xnorm*1.0d+4)/epp
        kmax=npt/(nthds*LB)

c  Electron concentration at the collector and emitter ohmic contacts
c       cn0c= nie(1,1)*exp(w(1,1)-w(3,1))
c       cn0e= nie(1,M)*exp(w(1,M)-w(3,M))

      xM1=par(ixm,M+1)
      xM =par(ixm,M)
      xM2=par(ixm,M-1)
      x2 =par(ixm,2)
      x1 =par(ixm,1)
      x0 =par(ixm,0)


      do 1 j=0,M+1
        cn(j)= 0.0
        do 2 iw=0,iwmax
          nptiw(iw,j)=0
cc        do 3 i=1,inmax
cc          nptnw(i,iw,j)=0
cc3       continue
```

```
 2        continue
 1      continue


        do k=1,nthds
          joutcaux(0,k) = 0
          jouteaux(0,k) = 0
        enddo

c$doacross local(k,j),
c$& shared(nthds,M,cnaux),
c$& mp_schedtype = simple
c$& , affinity(k) = data (cnaux(j,k))
        do k=1,nthds
          do j=0,M+1
            cnaux(j,k) = 0.0
          enddo
        enddo

c$doacross local(k,j,iw),
c$& shared(nthds,M,iwmax,nptiwaux,nptnwaux),
c$& mp_schedtype = simple
c$& , affinity(k) = data (nptiwaux(iw,j,k))
        do k=1,nthds
          do j=1,M+1
            do iw=0,iwmax
               nptiwaux(iw,j,k) = 0
cc             nptnwaux(n,iw,j,k) = 0
            enddo
          enddo
        enddo


        joc=0
        joe=0



c$doacross local(ntd,k,n,x,jp,iw,xj,xj1,cnj),
c$& shared(nthds,kmax,npt,LB,pt,ipt,ipi,wght,
c$&   nptiwaux,nptnwaux,par,cnaux,joutcaux,jouteaux),
c$& mp_schedtype = simple
c$& ,affinity(ntd) = thread ( ntd-1 )
c..$& ,affinity(ntd) = data ( pt (1, 1+LB*(ntd-1+(k-1)*nthds)))
        do 110 ntd=1,nthds
        do 100 k=0,kmax
        do 120 n=1+LB*(ntd-1+k*nthds),
     +           min(npt, LB*(ntd+k*nthds))

          x  = pt(ix,n)
          jp = ipt(icel,n)
          iw = ipt(iwgt,n)

          if (ipt(ival,n) .lt. 0) then
                  if (jp .lt. ipi(1)) then
                     joutcaux(0,ntd)=joutcaux(0,ntd)+1
                     joutcaux(joutcaux(0,ntd),ntd)=n
                  else
                     jouteaux(0,ntd)=jouteaux(0,ntd)+1
                     jouteaux(jouteaux(0,ntd),ntd)=n
                  endif
                  ipt(ival,n)=-ipt(ival,n)
          else
                  nptiwaux(iw,jp,ntd) = nptiwaux(iw,jp,ntd) + 1
                  nptnwaux(nptiwaux(iw,jp,ntd), iw, jp, ntd) = n
          endif


          xj = par(ixm,jp)
          xj1= par(ixm,jp-1)
          cnj= (x-xj1)/(xj-xj1)

          cnaux(jp-1,ntd) = cnaux(jp-1,ntd) + (1.-cnj) *wght(iw)
          cnaux(jp,ntd)   = cnaux(jp,ntd)   + cnj      *wght(iw)


 120    continue
 100    continue
 110    continue

        do k=1,nthds
         do 160 j=1,M
           cn(j) = cn(j) + cnaux(j,k)
 160    continue
        enddo


        do k=1,nthds
         do j=1,M+1
          do iw=0,iwmax
           do i=1,nptiwaux(iw,j,k)
            nptnw(nptiw(iw,j)+i,iw,j)=nptnwaux(i,iw,j,k)
           enddo
           nptiw(iw,j)=nptiw(iw,j)+nptiwaux(iw,j,k)
          enddo
         enddo
        enddo


        do k=1,nthds
         do j=1,joutcaux(0,k)
          joutc(joc+j)=joutcaux(j,k)
         enddo
         joc=joc+joutcaux(0,k)
        enddo

        do k=1,nthds
         do j=1,jouteaux(0,k)
          joute(joe+j)=jouteaux(j,k)
         enddo
         joe=joe+jouteaux(0,k)
        enddo


        jocd = joc
        joed = joe
        jel  = 0

        do 200 j=1,M
          cn(j) =cn(j) / (eppl*par(idx,j))
 200    continue
```

```
c  Correction related to the CIC method (half box integration; triangle shape):

c       j=ipi(1)
c       cn(j)=cn(j)+0.5*(cn(j)-cn(j-1))
c       j=ipi(1)+1
c       cn(j)=cn(j)-0.5*(cn(j+1)-cn(j))

c !!! Only for homojunctions in the BC interface
        j=ipi(1)
        cn(j)=0.5*(cn(j)+cn(j+1))
        cn(j+1)=cn(j)

c Valid for homo- & heterojunctions in the EB interface
        j=ipi(2)
        cn(j)=cn(j)+0.5*(cn(j)-cn(j-1))
        j=ipi(2)+1
        if ( cn(j) .gt. (0.5*(cn(j+1)-cn(j))) )
     +     cn(j)=cn(j)-0.5*(cn(j+1)-cn(j))


c!!! (Sequential: ntd=1)
        ntd=1

        if (cn(1).gt.cn0c) then
              if (jocd .ge. 1) then
230             continue
                jel=jel+1
                jelim(jel)=joutc(jocd)
                x=pt(ix,joutc(jocd))
                jp=ipt(icel,joutc(jocd))
                iw=ipt(iwgt,joutc(jocd))
                xj = par(ixm,jp)
                xj1= par(ixm,jp-1)
                cnj= (x-xj1)/(xj-xj1)
                cn(jp-1)=cn(jp-1)-(1.-cnj)/(eppl*par(idx,jp-1))*wght(iw)
                cn(jp)  =cn(jp)  -cnj/(eppl*par(idx,jp))*wght(iw)
                jocd = jocd -1
                if ( (cn(1).gt.cn0c) .and.
     +                                (jocd .ge. 1)) goto 230
              endif
              if (cn(1).gt.cn0c) then
               jp=1
               do iw=0,iwmax
                if ( (cn(1).gt.cn0c) .and.
     +                                (nptiw(iw,jp).ge.1)) then
231             continue
                jel=jel+1
                jelim(jel)=nptnw(nptiw(iw,jp),iw,jp)
                x=pt(ix,nptnw(nptiw(iw,jp),iw,jp))
                xj = par(ixm,jp)
                xj1= par(ixm,jp-1)
                cnj= (x-xj1)/(xj-xj1)
ccc             cn(jp-1)=cn(jp-1)-(1.-cnj)/(eppl*par(idx,jp-1))*wght(iw)
                cn(jp)  =cn(jp)  -cnj/(eppl*par(idx,jp))*wght(iw)
                nptiw(iw,jp) = nptiw(iw,jp) - 1
                if ( (cn(1).gt.cn0c) .and.
     +                                (nptiw(iw,jp).ge.1)) goto 231
                endif
               enddo
              endif
        endif
c ...without "else" in order to better adjust the concentration in cn(1)
        if (cn(1).lt.cn0c) then
          nnptc=1+NINT((cn0c-cn(1))*par(idx,1)*eppl)

          do 220 iw=0,iwmax
                niw=MOD(nnptc,base)
                nnptc=nnptc/base
                iv=1
                iregn=1
                jp=1

                do 221 i=1,niw

                npt=npt+1
                call create(tdt,iv,iregn,
     +            pt(ikx,npt),pt(iky,npt),pt(ikz,npt),
     +            pt(its,npt),pt(ix,npt),ivmax,iregnmax,
     +            efm,alf,gm,par,M,parNorm,LNorm,rn,ntd,nthds)

                ipt(ival,npt)=iv
                ipt(icel,npt)=jp
                ipt(iwgt,npt)=iw

                npt=npt+1
                call create(tdt,iv,iregn,
     +            pt(ikx,npt),pt(iky,npt),pt(ikz,npt),
     +            pt(its,npt),pt(ix,npt),ivmax,iregnmax,
     +            efm,alf,gm,par,M,parNorm,LNorm,rn,ntd,nthds)

                pt(ix,npt) = x1 - (pt(ix,npt-1)-x0)

                ipt(ival,npt)=iv
                ipt(icel,npt)=jp
                ipt(iwgt,npt)=iw

                nptiw(iw,jp) = nptiw(iw,jp) + 2
                nptnw(nptiw(iw,jp), iw, jp) = npt
                nptnw(nptiw(iw,jp)-1, iw, jp) = npt-1
221             continue

                cn(1) = cn(1) + niw *wght(iw) / (eppl*par(idx,1))

220       continue

        endif



        if (cn(M).gt.cn0e) then
              if (joed .ge. 1) then
330             continue
                jel=jel+1
                jelim(jel)=joute(joed)
                x=pt(ix,joute(joed))
                jp=ipt(icel,joute(joed))
                iw=ipt(iwgt,joute(joed))
                xj = par(ixm,jp)
                xj1= par(ixm,jp-1)
              cnj= (x-xj1)/(xj-xj1)
              cn(jp-1)=cn(jp-1)-(1.-cnj)/(epp1*par(idx,jp-1))*wght(iw)
              cn(jp)  =cn(jp)  -cnj/(epp1*par(idx,jp))*wght(iw)
              joed = joed -1
              if ( (cn(M).gt.cn0e) .and.
     +                                   (joed .ge. 1)) goto 330
            endif
            if (cn(M).gt.cn0e) then
              jp=M+1
              do iw=0,iwmax
               if ( (cn(M).gt.cn0e) .and.
     +                                  (nptiw(iw,jp).ge.1)) then
331            continue
               jel=jel+1
               jelim(jel)=nptnw(nptiw(iw,jp),iw,jp)
               x=pt(ix,nptnw(nptiw(iw,jp),iw,jp))
               xj = par(ixm,jp)
               xj1= par(ixm,jp-1)
               cnj= (x-xj1)/(xj-xj1)
              cn(jp-1)=cn(jp-1)-(1.-cnj)/(epp1*par(idx,jp-1))*wght(iw)
ccc            cn(jp)  =cn(jp)  -cnj/(epp1*par(idx,jp))*wght(iw)
               nptiw(iw,jp) = nptiw(iw,jp) - 1
               if ( (cn(M).gt.cn0e) .and.
     +                                  (nptiw(iw,jp).ge.1)) goto 331
               endif
              enddo
            endif
         endif
c ...without "else" in order to better adjust the concentration in cn(M)
         if (cn(M).lt.cn0e) then
           nnptc=1+NINT((cn0e-cn(M))*par(idx,M)*epp1)

           do 320 iw=0,iwmax
              niw=MOD(nnptc,base)
              nnptc=nnptc/base
              iv=1
              iregn=3
              jp=M+1

              do 321 i=1,niw

              npt=npt+1
              call create(tdt,iv,iregn,
     +          pt(ikx,npt),pt(iky,npt),pt(ikz,npt),
     +          pt(its,npt),pt(ix,npt),ivmax,iregnmax,
     +          efm,alf,gm,par,M,parNorm,LNorm,rn,ntd,nthds)

              ipt(ival,npt)=iv
              ipt(icel,npt)=jp
              ipt(iwgt,npt)=iw

              npt=npt+1
              call create(tdt,iv,iregn,
     +          pt(ikx,npt),pt(iky,npt),pt(ikz,npt),
     +          pt(its,npt),pt(ix,npt),ivmax,iregnmax,
     +          efm,alf,gm,par,M,parNorm,LNorm,rn,ntd,nthds)

              pt(ix,npt) = xM+(xM1 - pt(ix,npt-1))

              ipt(ival,npt)=iv
              ipt(icel,npt)=jp
              ipt(iwgt,npt)=iw

              nptiw(iw,jp) = nptiw(iw,jp) + 2
              nptnw(nptiw(iw,jp), iw, jp) = npt
              nptnw(nptiw(iw,jp)-1, iw, jp) = npt-1

321           continue

              cn(M)  = cn(M)   + niw *wght(iw) / (epp1*par(idx,M))

320        continue

         endif


       do j=1,M
         if (cn(j).lt.1.0d-15) cn(j)=1.0d-15
         w(3,j)=w(1,j)+par(16,j)-invfer(cn(j)*exp(par(16,j))/par(13,j))
       enddo


c   Append the particles not eliminated in joutc() and joute() to
c        nptiw() and nptnw()

         do j=1,jocd
           jp=ipt(icel,joutc(j))
           iw=ipt(iwgt,joutc(j))
           nptiw(iw,jp) = nptiw(iw,jp) + 1
           nptnw(nptiw(iw,jp), iw, jp) = joutc(j)
         enddo


         do j=1,joed
           jp=ipt(icel,joute(j))
           iw=ipt(iwgt,joute(j))
           nptiw(iw,jp) = nptiw(iw,jp) + 1
           nptnw(nptiw(iw,jp), iw, jp) = joute(j)
         enddo

c   Compact pt() and ipt() by supressing any eventual epmty position


         if (jel .gt. 1) call hpsort(jel,jelim)

         do i=jel,1,-1
           if (npt.ne.jelim(i)) then
                 pt(ikx,jelim(i))=pt(ikx,npt)
                 pt(iky,jelim(i))=pt(iky,npt)
                 pt(ikz,jelim(i))=pt(ikz,npt)
                 pt(its,jelim(i))=pt(its,npt)
                 pt(ix ,jelim(i))=pt(ix ,npt)
                 ipt(ival,jelim(i))=ipt(ival,npt)
                 ipt(icel,jelim(i))=ipt(icel,npt)
                 ipt(iwgt,jelim(i))=ipt(iwgt,npt)

                 j=nptiw(ipt(iwgt,npt),ipt(icel,npt))
350              if (nptnw(j,ipt(iwgt,npt),ipt(icel,npt)).ne.npt) then
                  j=j-1
                  goto 350
                 else
                  nptnw(j,ipt(iwgt,npt),ipt(icel,npt))=jelim(i)
            endif
         endif
         npt=npt-1
        enddo


c  Split particles that are too big for the concentration of the cell

        do 399 iregn=1,iregnmax

        if (iregn.eq.1) then
          i1=1
          i2=ipi(1)
        else if (iregn.eq.2) then
          i1=ipi(1)+2
          i2=ipi(2)
        else if (iregn.eq.3) then
          i1=ipi(2)+2
          i2=M+1
        else
          STOP 'more regions than expected in initiaMC()'
        endif


        do 400 j=i1,i2

          iwtop = 0
          do 410 iw=1,iwmax
                if ( nptiw(iw,j) .ge. nptiw(iwtop,j) ) iwtop=iw
410       continue

          do 420 iw=iwmax, iwtop+3, -1
             do 430 k=1,nptiw(iw,j)
                kx= pt (ikx, nptnw(k, iw, j) )
                ky= pt (iky, nptnw(k, iw, j) )
                kz= pt (ikz, nptnw(k, iw, j) )
                ts= pt (its, nptnw(k, iw, j) )
                x = pt (ix , nptnw(k, iw, j) )

                iv= ipt (ival, nptnw(k, iw, j) )
                jp= ipt (icel, nptnw(k, iw, j) )
                ipt (iwgt, nptnw(k, iw, j) ) = iw -1

                nptnw(nptiw(iw-1,j)+base*k, iw-1, j) =
     +                                  nptnw(k, iw, j)
                do 440 i=1,base1
                  n = npt + i
                  nptnw(nptiw(iw-1,j)+base*(k-1)+i, iw-1, j) = n
                  pt(ikx,n)=kx
                  pt(iky,n)=ky
                  pt(ikz,n)=kz
                  pt(its,n)=ts
                  pt(ix,n) =x
                  ipt(ival,n)=iv
                  ipt(icel,n)=jp
                  ipt(iwgt,n)=iw -1
440             continue

                npt = npt + base1

430          continue
                nptiw(iw-1,j) = nptiw(iw-1,j) + base * nptiw(iw,j)
                nptiw(iw,j) = 0

420         continue
400       continue
399       continue


c  Group/split particles in order to limit the total amount of particles

        lfree = 0

        do 499 iregn=1,iregnmax

        if (iregn.eq.1) then
          i1=1
          i2=ipi(1)
        else if (iregn.eq.2) then
          i1=ipi(1)+2
          i2=ipi(2)
        else if (iregn.eq.3) then
          i1=ipi(2)+2
          i2=M+1
        else
          STOP 'more regions than expected in initiaMC()'
        endif


        do 500 j=i1,i2

          npti(j) = 0
          do 501 iw = 0, iwmax
                npti(j) = npti(j) + nptiw(iw,j)
501       continue
          if (npti(j).le.0) STOP 'Not even a particle in the cell'
c!! test:
          if (npti(j) .gt. nptitop) nptitop = npti(j)
c         if (lfree .gt. lfrtop) lfrtop = lfree
c         if (npt .gt. npttop) npttop = npt

          iw = 0
          incr=0
502       if ( npti(j) .gt. npcmax-incr ) then
                incr=increm
503             if ( nptiw(iw,j) .ge. base ) then

                  nptiw(iw+1,j) = nptiw(iw+1,j) + 1
                  nptnw(nptiw(iw+1,j), iw+1, j) =
     +                              nptnw(nptiw(iw,j), iw, j)
                  nptiw(iw,j) = nptiw(iw,j) - base
                  npti(j) = npti(j) - base1

                  x=0
                  lfree = lfree + 1
                  do 504 i=1,base1
                    ifree(i,lfree) = nptnw(nptiw(iw,j)+i, iw, j)
                    x = x + pt (ix , nptnw(nptiw(iw,j)+i, iw, j) )
504               continue
                    x = x + pt (ix , nptnw(nptiw(iw,j)+base, iw, j) )
                    pt (ix , nptnw(nptiw(iw+1,j), iw+1, j) )= x*basei
```

```
              ipt (iwgt , nptnw(nptiw(iw+1,j), iw+1, j) )= iw+1
          else

            iw = iw +1
            if (iw.gt.iwmax) then
             print*, 'No particles to group! t+dt,j,iw=',
     +                             tdt,j,iw
             STOP
            endif
          goto 503
          endif

      goto 502
      else

      iw = iwmax
      incr=0
520   if ( npti(j) .lt. npcmin+incr ) then
          incr=increm
521       if ( nptiw(iw,j) .eq. 0 ) then
            iw = iw -1
            if (iw.lt.0) then
             print*, 'No particles to split! t+dt,j,iw=',
     +                             tdt,j,iw
             STOP
            endif
          goto 521
          endif

          kx= pt (ikx, nptnw(nptiw(iw,j), iw, j) )
          ky= pt (iky, nptnw(nptiw(iw,j), iw, j) )
          kz= pt (ikz, nptnw(nptiw(iw,j), iw, j) )
          ts= pt (its, nptnw(nptiw(iw,j), iw, j) )
          x = pt (ix , nptnw(nptiw(iw,j), iw, j) )

          iv= ipt (ival, nptnw(nptiw(iw,j), iw, j) )
          jp= ipt (icel, nptnw(nptiw(iw,j), iw, j) )
          ipt (iwgt, nptnw(nptiw(iw,j), iw, j) ) = iw -1

          nptnw(nptiw(iw-1,j)+base, iw-1, j) =
     +                         nptnw(nptiw(iw,j), iw, j)
          nptiw(iw,j) = nptiw(iw,j) - 1
          npti(j) = npti(j) + base1

          if (lfree .gt. 0) then
            do i=1,base1
                  n = ifree (i,lfree)
                  nptnw(nptiw(iw-1,j)+i, iw-1, j) = n
                  pt(ikx,n)=kx
                  pt(iky,n)=ky
                  pt(ikz,n)=kz
                  pt(its,n)=ts
                  pt(ix,n) =x
                  ipt(ival,n)=iv
                  ipt(icel,n)=jp
                  ipt(iwgt,n)=iw -1
            enddo

            lfree = lfree - 1

          else
            do i=1,base1
                  n = npt + i
                  nptnw(nptiw(iw-1,j)+i, iw-1, j) = n
                  pt(ikx,n)=kx
                  pt(iky,n)=ky
                  pt(ikz,n)=kz
                  pt(its,n)=ts
                  pt(ix,n) =x
                  ipt(ival,n)=iv
                  ipt(icel,n)=jp
                  ipt(iwgt,n)=iw -1
            enddo

            npt = npt + base1

          endif

          nptiw(iw-1,j) = nptiw(iw-1,j) + base

        goto 520
        endif

      endif

c!! test:
c         if (npti(j) .gt. nptitop) nptitop = npti(j)
          if (lfree .gt. lfrtop) lfrtop = lfree
c         if (npt .gt. npttop) npttop = npt

500   continue
499   continue

c!! test:
c         if (npti(j) .gt. nptitop) nptitop = npti(j)
c         if (lfree .gt. lfrtop) lfrtop = lfree
          if (npt .gt. npttop) npttop = npt


c  Compact pt() and ipt() by suppressing any eventual empty position
      if (lfree .gt. 0) call hpsort(lfree*base1,ifree)
      do 600 j=lfree,1,-1
        do 600 i=base1,1,-1
              n = ifree (i,j)
              pt(ikx,n)=pt(ikx,npt-base1*(lfree-j+1)+i)
              pt(iky,n)=pt(iky,npt-base1*(lfree-j+1)+i)
              pt(ikz,n)=pt(ikz,npt-base1*(lfree-j+1)+i)
              pt(its,n)=pt(its,npt-base1*(lfree-j+1)+i)
              pt(ix,n) =pt(ix ,npt-base1*(lfree-j+1)+i)
              ipt(ival,n)=ipt(ival,npt-base1*(lfree-j+1)+i)
              ipt(icel,n)=ipt(icel,npt-base1*(lfree-j+1)+i)
              ipt(iwgt,n)=ipt(iwgt,npt-base1*(lfree-j+1)+i)
c       enddo
c     enddo
600   continue

      npt = npt - lfree * base1
c     lfree = 0

    return
    end
```

```
c************************ subroutine hpsort **************************
c--------------------------------------------------------------------
c
c
c                  by Pau Garcias i Salva'. Nov., 1998
c                (Numerical Recipes in FORTRAN, 2nd.ed.)
c
c--------------------------------------------------------------------
      subroutine hpsort(n,ra)
c
      implicit none
c
          integer
     +        n,ra(n)
          integer
     +        i,ir,j,l,rra


      if (n .lt. 2) return

      l=n/2+1
      ir=n

10    continue

        if (l .gt. 1) then
              l=l-1
              rra=ra(l)
        else
              rra=ra(ir)
              ra(ir)=ra(1)
              ir=ir-1
              if (ir .eq. 1) then
                 ra(1)=rra
                 return
              endif
        endif

        i=l
        j=l+l

20      if (j .le. ir) then
              if (j .lt. ir) then
                      if (ra(j) .lt. ra(j+1)) j=j+1
              endif
              if (rra .lt. ra(j)) then
                      ra(i)=ra(j)
                      i=j
                      j=j+j
              else
                      j=ir+1
              endif
        goto 20
        endif

        ra(i)=rra

      goto 10

      end
```

```
c************************ subroutine create **************************
c--------------------------------------------------------------------
c
c
c                  by Pau Garcias i Salva'. Nov., 1997
c
c--------------------------------------------------------------------
      subroutine create(tdt,iv,iregn,kx,ky,kz,ts,x,ivmax,iregnmax,
     +        efm,alf,gm,par,M,parNorm,LNorm,rn,ntd,nthds)
c
c     storage allocation
c
c          implicit double precision (a-h,o-z)
c          implicit integer (i-n)

      implicit none
c

c  parameters to limit the value of the initial energy of the new particles
c   to a reasonable value of 0.5eV
c    ei=-1.5*log(rmin+rmin1*rnd())
c
c    rnd()=]0.0 ... 1.0[,    rmin,   rmin1=1-rmin

          double precision
     +        rmin,rmin1,rmin2
          parameter(rmin=1.0d-200)
          parameter(rmin1=1.0d+00-rmin)

c  the same, in order to limit the free flight duration to a reasonable value
          parameter(rmin2=1.0d-50)


          integer
     +        iv,iregn,M,ivmax,iregnmax,LNorm,ntd,nthds
          double precision
     +        tdt,kx,ky,kz,ts,x,
     +        efm(ivmax,iregnmax),alf(ivmax,iregnmax),
     +        gm(ivmax,iregnmax),
     +        par(19,0:M+1),parNorm(LNorm)
          real*4
     +        rn(0:2+5+24,nthds)


          double precision
     +        efm1,alf1,gm1,ei,ak,cb,sb,fai,He,pi


      He = parNorm(20)
      pi = parNorm(30)

      call rcarry(rn(0,ntd),5)

      iv=1
      efm1=efm(iv,iregn)
      alf1=alf(iv,iregn)
      gm1=-1./gm(iv,iregn)
```

```
         ei=-1.5*log(rmin+rmin1*rn(1,ntd))
         ak=sqrt(efm1*ei*(1.+alfl*ei)/He)

         cb=rn(2,ntd)
         sb=sqrt(1.-cb*cb)
         fai=2.*pi*rn(3,ntd)

         kx=ak*cb
         ky=ak*sb*cos(fai)
         kz=ak*sb*sin(fai)
         ts=tdt+gm1*log(rmin2+(1.0-rmin2)*rn(4,ntd))

      if (iregn.eq.1) then
         x=par(19,0)-par(18,0)*rn(5,ntd)
      else
         x=par(19,M+1)-par(18,M+1)*rn(5,ntd)
         kx=-kx
      endif

      return
      end

c************************* subroutine field ***************************
c----------------------------------------------------------------------
c
c
c                   by Pau Garcias i Salva'. Nov., 1997
c
c----------------------------------------------------------------------
        subroutine field(gradp,w,par,ipi,M,nbmax)
c
c       storage allocation
c
c            implicit double precision (a-h,o-z)
c            implicit integer (i-n)

        implicit none
c
           integer
     +           ipi(2),M,nbmax
           double precision
     +           gradp(0:M+1),w(nbmax,M),par(19,0:M+1)

           integer
     +           j


      do 10 j=2,M-1
        gradp(j)=(w(1,j+1)-w(1,j-1))/(par(1,j+1)-par(1,j-1))
10    continue

cc Assuming ohmic contacts --> charge neutrality: ro=0 --> Poisson eq.
cc  (See also, K. Tomizawa, 1993, p. 281 (book) )

      gradp(1)=gradp(2)
      gradp(0)=gradp(2)

      gradp(M)  =gradp(M-1)
      gradp(M+1)=gradp(M-1)

cc field --> Laux & Fischetti,1991
```

```
      j=ipi(1)
      gradp(j)=((w(1,j)  -w(1,j-1))/(par(1,j  )-par(1,j-1))*par(15,j) +
     +          (w(1,j+2)-w(1,j+1))/(par(1,j+2)-par(1,j+1))*par(15,j+1))
     +          /(2.0*par(15,j))

      gradp(j+1)=gradp(j)*par(15,j)/par(15,j+1)


      j=ipi(2)
      gradp(j)=((w(1,j)  -w(1,j-1))/(par(1,j  )-par(1,j-1))*par(15,j) +
     +          (w(1,j+2)-w(1,j+1))/(par(1,j+2)-par(1,j+1))*par(15,j+1))
     +          /(2.0*par(15,j))

      gradp(j+1)=gradp(j)*par(15,j)/par(15,j+1)

      return
      end

c----------------------------------------------------------------------
ceeee+i**#**** ****#**** ****#**** ****#**** ****#**** ****#**** ****#72------80
c----------------------------------------------------------------------
```

```
c*********************** subroutine DDholes ***********************
c-------------------------------------------------------------------
c
c
c                  by Pau Garcias i Salva'. Nov., 1997
c
c-------------------------------------------------------------------
       subroutine DDholes (PARINT,T,IPI,IMI,IBT,
     + M,W,PAR,NIE,S,B,C,MP,NBMAX,NMAX,pdt0,dt,
     + NTMAX,ITERM,LPROX,ERROR,CORRE,ITER,CORL,TIP,WI,FET)
c
      implicit none
c
      INTEGER
     + IPI(2),IMI(2),NBMAX,NMAX,NTMAX,IBT,ITERM,ITER,TIP,M

      DOUBLE PRECISION
     + T(3),MP,
     + W(NBMAX,NMAX),PAR(19,0:NMAX+1),S(NTMAX),
     + B(NMAX),PROX,PROXA,COR,CORA,C(NMAX),
     + NIE(2,NMAX),PARINT(5,2),CORRE,LPROX,ERROR,CORL,
     + FNC,FNV,FERM12,WI(1,NMAX),FET(5,4),
     + pdt0(M),dt

      INTEGER
     + J

c  compute the initial concentration of holes, pdt0(J), before the
c   new time step dt is applied.

        DO 1 J=1,M
         FNV = (W(2,J)-W(1,J)) + PAR(17,J)
         NIE(2,J) = PAR(14,J)*FERM12(FNV)/EXP(FNV)
         pdt0(J)=NIE(2,J)*EXP(W(2,J)-W(1,J))
1       CONTINUE

c       inilializamos algunos parametros utilizados

      ITER=0
      ERROR=0
      CORRE=0
      TIP=0
      PROX= 1.0D+10
      COR = 1.0D+10

c       ******************************************************
c       ******************************************************
c       COMIENZA EL PROCESO ITERATIVO
C       *****************************************************

130     CONTINUE

        ITER=ITER+1

c       ****************************************
c       actualizamos los valores de proxa y cora
c       ****************************************
        PROXA=PROX
        CORA=COR

c       ****************************************
c       actualizamos los valores de nien y niep
c       ****************************************


        DO 2500 J=1,M
         FNC = (W(1,J)-W(3,J)) + PAR(16,J)
         FNV = (W(2,J)-W(1,J)) + PAR(17,J)
         NIE(1,J) = PAR(13,J)*FERM12(FNC)/EXP(FNC)
         NIE(2,J) = PAR(14,J)*FERM12(FNV)/EXP(FNV)
2500   CONTINUE


c       ******************************************
c       calculamos unos nuevos valores de S y B
c       ******************************************

        CALL SETMATh(M,IPI,IMI,IBT,S,B,pdt0,dt,
     +  T,W,WI,PAR,PARINT,FET,NIE,NBMAX,NTMAX)

c       *****************************************
c       calculo de proximidad a cero
c       *****************************************

        CALL RHSAV(PROX,B,M)

c       ****************************************************
c       solucionamos el sistema de ecuaciones para hallar dx
c       ****************************************************

      CALL LINSOL(IPI,S,B,C,M,MP,NBMAX,NMAX,NTMAX)

c       ************************************
c       calculo de los errores en dx
c       ************************************

      CALL RHSAV(COR,C,M)


      DO 220 J=1,M
        W(2,J)=W(2,J)+C(J)
220     CONTINUE


c       ******************************************
c       ******************************************
c       si la nueva solucion es la exacta salimos
c       ******************************************

        IF(PROX.LT.LPROX.AND.COR.LT.CORL) THEN
                TIP=1
                ERROR=PROX
                CORRE=COR
                RETURN

c       ****************************************************
c       ****************************************************
```

```
c       si la nueva solucion mejora la anterior, permitimos
c       seguir iterando para mejorarla aun mas
c       ****************************************************

        ELSE IF((PROX.LT.PROXA.OR.PROX.LT.1D-10).
     +           AND.COR.LT.CORA) THEN
                GOTO 130


c       ******************************************************
c       ******************************************************
c       si el numero de iteraciones supera el maximo permitido
c       salimos indicando que no converge (TIP = 2)
c       ******************************************************

        ELSE IF(ITER.GT.ITERM) THEN
                TIP=2
                ERROR=PROX
                CORRE=COR
                RETURN

c       *****************************************************
c       *****************************************************
c       si la nueva solucion hace f1,f2,f3 aprox cero
c       pero dx crece, reducimos el valor de dx en aquellos
c       puntos que ha aumentado demasiado y calculamos los w
c       *****************************************************

        ELSE IF(COR.GE.CORA.OR.(PROX.GE.1D-10.AND.COR.LT.CORA)) THEN
                GOTO 130

        END IF

c       ***************************************************
c       errores cometidos en la simulacion numerica
c       ***************************************************
      TIP=1
      ERROR=PROX
      CORRE=COR
      RETURN
      END



c************************ subroutine SETMATh ************************
c--------------------------------------------------------------------
c
        SUBROUTINE SETMATh(M,IPI,IMI,IBT,S,B,pdt0,dt,
     +  T,WP,WI,PAR,PARINT,FET,NIE,NBMAX,NTMAX)
c
      implicit none
c

      INTEGER
     + M,IPI(2),IMI(2),IBT,NBMAX,NTMAX

      DOUBLE PRECISION
     + S(NTMAX),B(M),pdt0(M),dt,
     + T(3),WP(NBMAX,M),WI(1,M),PAR(19,0:M+1),PARINT(5,2),
     + FET(5,4),NIE(2,M)

      INTEGER
     + IPL,IPU,IPE,J

      DOUBLE PRECISION
     + CP,CM,XI,XIM,BI,BIM,RAB,RS,DRAB,DRS,IR,
     + EFPP2,EFPP1,EFPP,EFP,EFPM,EPSP,EPS,EPSM,PIP,PI,PIM,
     + DIR(2),DRI(2),RI

C       pointers for one-dimensional array s
      IPL=M-1
      IPU=2*M-1
      IPE=3*M-2


C       setting up s and b matrices
c   General case:

      DO 10 J=2,IPI(1)-1

         CP=(PAR(4,J+1)*NIE(2,J+1)+PAR(4,J)*NIE(2,J))/
     +           (2.*(PAR(1,J+1)-PAR(1,J)))
         CM=(PAR(4,J)*NIE(2,J)+PAR(4,J-1)*NIE(2,J-1))/
     +           (2.*(PAR(1,J)-PAR(1,J-1)))
         XI =WP(1,J+1)-WP(1,J)
         XIM=WP(1,J)-WP(1,J-1)

         CALL BER(XI,BI)
         CALL BER(XIM,BIM)
         CALL RPIG(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
         IR=PAR(18,J)*(RAB+RS)

         EFPP=EXP(WP(2,J+1))
         EFP=EXP(WP(2,J))
         EFPM=EXP(WP(2,J-1))
         EPS=EXP(WP(1,J))
         EPSM=EXP(WP(1,J-1))


        B(J)=CP*(EFPP-EFP)*BI/EPS - CM*(EFP-EFPM)*BIM/EPSM - IR
        S(J)=CP*EFP*BI/EPS+CM*EFP*BIM/EPSM+
     +          PAR(18,J)*(DRAB+DRS)
        S(IPL+J)=-CM*EFPM*BIM/EPSM
        S(IPU+J)=-CP*EFPP*BI /EPS

10    CONTINUE


      DO 20 J=IPI(1)+2,IPI(2)-1

         CP=(PAR(4,J+1)*NIE(2,J+1)+PAR(4,J)*NIE(2,J))/
     +           (2.*(PAR(1,J+1)-PAR(1,J)))
         CM=(PAR(4,J)*NIE(2,J)+PAR(4,J-1)*NIE(2,J-1))/
     +           (2.*(PAR(1,J)-PAR(1,J-1)))
         XI =WP(1,J+1)-WP(1,J)
         XIM=WP(1,J)-WP(1,J-1)

         CALL BER(XI,BI)
         CALL BER(XIM,BIM)
```

```
       CALL RPIG(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
       IR=PAR(18,J)*(RAB+RS)

       EFPP=EXP(WP(2,J+1))
       EFP=EXP(WP(2,J))
       EFPM=EXP(WP(2,J-1))
       EPS=EXP(WP(1,J))
       EPSM=EXP(WP(1,J-1))


      B(J)=CP*(EFPP-EFP)*BI/EPS - CM*(EFP-EFPM)*BIM/EPSM - IR
      S(J)=CP*EFP*BI/EPS+CM*EFP*BIM/EPSM+
     +        PAR(18,J)*(DRAB+DRS)
      S(IPL+J)=-CM*EFPM*BIM/EPSM
      S(IPU+J)=-CP*EFPP*BI /EPS

20    CONTINUE


      DO 30 J=IPI(2)+2,M-1

       CP=(PAR(4,J+1)*NIE(2,J+1)+PAR(4,J)*NIE(2,J))/
     +         (2.*(PAR(1,J+1)-PAR(1,J)))
       CM=(PAR(4,J)*NIE(2,J)+PAR(4,J-1)*NIE(2,J-1))/
     +         (2.*(PAR(1,J)-PAR(1,J-1)))
       XI =WP(1,J+1)-WP(1,J)
       XIM=WP(1,J)-WP(1,J-1)

       CALL BER(XI,BI)
       CALL BER(XIM,BIM)
       CALL RPIG(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
       IR=PAR(18,J)*(RAB+RS)

       EFPP=EXP(WP(2,J+1))
       EFP=EXP(WP(2,J))
       EFPM=EXP(WP(2,J-1))
       EPS=EXP(WP(1,J))
       EPSM=EXP(WP(1,J-1))


      B(J)=CP*(EFPP-EFP)*BI/EPS - CM*(EFP-EFPM)*BIM/EPSM - IR
      S(J)=CP*EFP*BI/EPS+CM*EFP*BIM/EPSM+
     +        PAR(18,J)*(DRAB+DRS)
      S(IPL+J)=-CM*EFPM*BIM/EPSM
      S(IPU+J)=-CP*EFPP*BI /EPS

30    CONTINUE

c  Corrections to the general case:
c   a)  1,ibt,M; 2,M-1
c   a1) 1,ibt,M

      J=1
      B(J) = T(1)-WP(2,1)
      S(J) = 1.
      S(IPU+J) = 0.


      J=IBT
      B(J) = T(2)-WP(2,J)
      S(J) = 1.
      S(IPL+J) = 0.
      S(IPU+J) = 0.

      J=M
      B(J) = T(3)-WP(2,J)
      S(J) = 1.
      S(IPL+J) = 0.


c   a2) 2,M-1

      J=2
      CALL RPIG(J-1,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
      B(J)=B(J)-0.5*PAR(1,J)*(RAB+RS)
      S(IPL+J) = S(IPL+J) +0.5*PAR(1,J)*(DRAB+DRS)

      J=M-1
      CALL RPIG(J+1,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
      B(J)=B(J)-0.5*(PAR(1,J+1)-PAR(1,J))*(RAB+RS)
      S(IPU+J) = S(IPU+J) +0.5*(PAR(1,J+1)-PAR(1,J))*(DRAB+DRS)



c  Corrections to the general case:
c   b) ipi(2), ipi(2)+1

      if ( (imi(2).eq.2) .or. (imi(2).eq.3) ) then
c   b1) ipi(2) when imi(2)= 2 or 3

       J=IPI(2)
       CM=(PAR(4,J)*NIE(2,J)+PAR(4,J-1)*NIE(2,J-1))/
     +         (2.*(PAR(1,J)-PAR(1,J-1)))
       XIM=WP(1,J)-WP(1,J-1)

       CALL BER(XIM,BIM)

       EFP=EXP(WP(2,J))
       EFPM=EXP(WP(2,J-1))
       EPSM=EXP(WP(1,J-1))
       PIP=NIE(2,J+1)*EXP(WP(2,J+1)-WP(1,J+1))
       PI=NIE(2,J)*EXP(WP(2,J)-WP(1,J))

      B(J)=-FET(3,2)*(PIP-PI*FET(4,2)*FET(5,2))
     +         +CM*(EFP-EFPM)*BIM/EPSM
      S(J)=-FET(3,2)*PI*FET(4,2)*FET(5,2)-CM*EFP*BIM/EPSM
      S(IPL+J)=CM*EFPM*BIM/EPSM
      S(IPU+J)=FET(3,2)*PIP
      S(IPE+4)= 0.


c   b2) ipi(2)+1 when imi(2)= 2 or 3

       J=IPI(2)+1
       CP=(PAR(4,J+1)*NIE(2,J+1)+PAR(4,J)*NIE(2,J))/
     +         (2.*(PAR(1,J+1)-PAR(1,J)))
       XI=WP(1,J+1)-WP(1,J)

       CALL BER(XI,BI)
      CALL RPIG(J-1,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
      IR=PAR(18,J-1)*(RAB+RS)
      DIR(1)=PAR(18,J-1)*(DRAB+DRS)

      CALL RPII(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX,
     +                RI,DRI,IPI,PARINT)
      IR=IR+PAR(18,J)*(RAB+RS)+RI
      DIR(1)=DIR(1)+DRI(2)
      DIR(2)=PAR(18,J)*(DRAB+DRS)+DRI(1)

       EFPP=EXP(WP(2,J+1))
       EFP=EXP(WP(2,J))
       EPS=EXP(WP(1,J))
       PI=NIE(2,J)*EXP(WP(2,J)-WP(1,J))
       PIM=NIE(2,J-1)*EXP(WP(2,J-1)-WP(1,J-1))

      B(J)=-FET(3,2)*(PI-PIM*FET(4,2)*FET(5,2))
     +        +CP*(EFPP-EFP)*BI/EPS - IR
      S(J)=FET(3,2)*PI + CP*EFP*BI/EPS+DIR(2)
      S(IPL+J)=-FET(3,2)*PIM*FET(4,2)*FET(5,2)+DIR(1)
      S(IPU+J)=-CP*EFPP*BI/EPS
      S(IPE+3)= 0.

      else
c   b3) ipi(2) when imi(2)= 1 or 4

       J=IPI(2)
       CP=(PAR(4,J+2)*NIE(2,J+2)+PAR(4,J+1)*NIE(2,J+1))/
     +            (2.*(PAR(1,J+2)-PAR(1,J+1)))
       CM=(PAR(4,J)*NIE(2,J)+PAR(4,J-1)*NIE(2,J-1))/
     +            (2.*(PAR(1,J)-PAR(1,J-1)))
       XI=WP(1,J+2)-WP(1,J+1)
       XIM=WP(1,J)-WP(1,J-1)

       CALL BER(XI,BI)
       CALL BER(XIM,BIM)
      CALL RPIG(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
      IR=PAR(18,J)*(RAB+RS)
      DIR(1)=PAR(18,J)*(DRAB+DRS)

      CALL RPII(J+1,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX,
     +                RI,DRI,IPI,PARINT)
      IR=IR+PAR(18,J+1)*(RAB+RS)+RI
      DIR(1)=DIR(1)+DRI(2)
      DIR(2)=PAR(18,J+1)*(DRAB+DRS)+DRI(1)


       EFPP2=EXP(WP(2,J+2))
       EFPP1=EXP(WP(2,J+1))
       EFP=EXP(WP(2,J))
       EFPM=EXP(WP(2,J-1))
       EPSP=EXP(WP(1,J+1))
       EPS=EXP(WP(1,J))
       EPSM=EXP(WP(1,J-1))

      B(J)=CP*(EFPP2-EFPP1)*BI/EPSP-CM*(EFP-EFPM)*BIM/EPSM-IR
      S(J)=CM*EFP*BIM/EPSM + DIR(1)
      S(IPL+J)=-CM*EFPM*BIM/EPSM
      S(IPU+J)=CP*EFPP1*BI/EPSP + DIR(2)
      S(IPE+4)=-CP*EFPP2*BI/EPSP

c   b4) ipi(2)+1 when imi(2)= 1 or 4

       J=IPI(2)+1
       PIP=NIE(2,J)*EXP(WP(2,J)-WP(1,J))
       PI=NIE(2,J-1)*EXP(WP(2,J-1)-WP(1,J-1))

      B(J)=PIP-PI
      S(J)=-PIP
      S(IPL+J)=PI
      S(IPU+J)= 0.
      S(IPE+3)= 0.

      endif


c  Corrections to the general case:
c   c) ipi(1), ipi(1)+1  when imi(1)= 2 or 3

      if ( (imi(1).eq.2) .or. (imi(1).eq.3) ) then

              stop 'Case not modelled: imi(1)= 2 or 3'
      else
c   c1) ipi(1) when imi(1)= 1 or 4

       J=IPI(1)
       CP=(PAR(4,J+2)*NIE(2,J+2)+PAR(4,J+1)*NIE(2,J+1))/
     +            (2.*(PAR(1,J+2)-PAR(1,J+1)))
       CM=(PAR(4,J)*NIE(2,J)+PAR(4,J-1)*NIE(2,J-1))/
     +            (2.*(PAR(1,J)-PAR(1,J-1)))
       XI=WP(1,J+2)-WP(1,J+1)
       XIM=WP(1,J)-WP(1,J-1)

       CALL BER(XI,BI)
       CALL BER(XIM,BIM)
      CALL RPIG(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
      IR=PAR(18,J)*(RAB+RS)
      DIR(1)=PAR(18,J)*(DRAB+DRS)

      CALL RPII(J+1,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX,
     +                RI,DRI,IPI,PARINT)
      IR=IR+PAR(18,J+1)*(RAB+RS)+RI
      DIR(1)=DIR(1)+DRI(2)
      DIR(2)=PAR(18,J+1)*(DRAB+DRS)+DRI(1)


       EFPP2=EXP(WP(2,J+2))
       EFPP1=EXP(WP(2,J+1))
       EFP=EXP(WP(2,J))
       EFPM=EXP(WP(2,J-1))
       EPSP=EXP(WP(1,J+1))
       EPS=EXP(WP(1,J))
       EPSM=EXP(WP(1,J-1))

      B(J)=CP*(EFPP2-EFPP1)*BI/EPSP-CM*(EFP-EFPM)*BIM/EPSM-IR
      S(J)=CM*EFP*BIM/EPSM + DIR(1)
      S(IPL+J)=-CM*EFPM*BIM/EPSM
      S(IPU+J)=CP*EFPP1*BI/EPSP + DIR(2)
      S(IPE+2)=-CP*EFPP2*BI/EPSP

c   c2) ipi(1)+1 when imi(1)= 1 or 4

       J=IPI(1)+1
       PIP=NIE(2,J)*EXP(WP(2,J)-WP(1,J))
```

```
          PI=NIE(2,J-1)*EXP(WP(2,J-1)-WP(1,J-1))

          B(J)=PIP-PI
          S(J)=-PIP
          S(IPL+J)=PI
          S(IPU+J)= 0.
          S(IPE+1)= 0.

          endif

          if (dt .gt. 0.0) call TRANS(M,IPI,IMI,S,B,pdt0,dt,
     +    WP,PAR,NIE,NBMAX,NTMAX)

        RETURN
        END




c************************ subroutine TRANS **************************
c--------------------------------------------------------------------
c
          SUBROUTINE TRANS(M,IPI,IMI,S,B,pdt0,dt,
     +    WP,PAR,NIE,NBMAX,NTMAX)
c
        implicit none
c

        INTEGER
     + M,IPI(2),IMI(2),NBMAX,NTMAX

        DOUBLE PRECISION
     + S(NTMAX),B(M),pdt0(M),dt,
     + WP(NBMAX,M),PAR(19,0:M+1),
     + NIE(2,M)

        INTEGER
     + IPL,IPU,J

        DOUBLE PRECISION
     + pdt,pdt1,idp

C          pointers for one-dimensional array s
        IPL=M-1
        IPU=2*M-1


C          setting up s and b matrices
c    General case:

        DO 10 J=2,IPI(1)-1

          pdt= nie(2,j)*exp(wp(2,j)-wp(1,j))
          idp= par(18,j)* (pdt-pdt0(j))/dt

          B(J)=B(J) - idp
          S(J)=S(J) + par(18,j)* pdt/dt

10      CONTINUE

        DO 20 J=IPI(1)+2,IPI(2)-1

          pdt= nie(2,j)*exp(wp(2,j)-wp(1,j))
          idp= par(18,j)* (pdt-pdt0(j))/dt

          B(J)=B(J) - idp
          S(J)=S(J) + par(18,j)* pdt/dt

20      CONTINUE

        DO 30 J=IPI(2)+2,M-1

          pdt= nie(2,j)*exp(wp(2,j)-wp(1,j))
          idp= par(18,j)* (pdt-pdt0(j))/dt

          B(J)=B(J) - idp
          S(J)=S(J) + par(18,j)* pdt/dt

30      CONTINUE


c    Corrections to the general case:
c     a)  1,ibt,M; 2,M-1
c     a1) 1,ibt,M

c     no changes are needed

c     a2) 2,M-1

          J=2
          pdt= nie(2,j-1)*exp(wp(2,j-1)-wp(1,j-1))
          idp= 0.5*par(1,j)* (pdt-pdt0(j-1))/dt
          B(J)=B(J) - idp
          S(IPL+J)=S(IPL+J) + 0.5*par(1,j)* pdt/dt

          J=M-1
          pdt= nie(2,j+1)*exp(wp(2,j+1)-wp(1,j+1))
          idp= 0.5*(par(1,j+1)-par(1,j))* (pdt-pdt0(j+1))/dt
          B(J)=B(J) - idp
          S(IPU+J)=S(IPU+J) + 0.5*(par(1,j+1)-par(1,j))* pdt/dt


c    Corrections to the general case:
c     b) ipi(2), ipi(2)+1

          if ( (imi(2).eq.2) .or. (imi(2).eq.3) ) then
c     b1) ipi(2) when imi(2)= 2 or 3

c          J=IPI(2) --> no correction is needed


c     b2) ipi(2)+1 when imi(2)= 2 or 3

           J=IPI(2)+1
          pdt = nie(2,j)   *exp(wp(2,j)   -wp(1,j))
          pdt1= nie(2,j-1)*exp(wp(2,j-1)-wp(1,j-1))
          idp= (par(18,j)   * (pdt-pdt0(j))    +
     +            par(18,j-1)* (pdt1-pdt0(j-1)) )/dt
```

```
         B(J)=B(J)  - idp
         S(J)=S(J)  + par(18,j)* pdt/dt
         S(IPL+J)=S(IPL+J) + par(18,j-1)* pdt1/dt

         else
c    b3) ipi(2) when imi(2)= 1 or 4

          J=IPI(2)
         pdt = nie(2,j)   *exp(wp(2,j)   -wp(1,j))
         pdt1= nie(2,j+1)*exp(wp(2,j+1)-wp(1,j+1))
         idp= (par(18,j)   * (pdt-pdt0(j))    +
     +         par(18,j+1)* (pdt1-pdt0(j+1)) )/dt

         B(J)=B(J)  - idp
         S(J)=S(J)  + par(18,j)* pdt/dt
         S(IPU+J)=S(IPU+J) + par(18,j+1)* pdt1/dt


c    b4) ipi(2)+1 when imi(2)= 1 or 4

c         J=IPI(2)+1 --> no correction is needed

         endif


c   Corrections to the general case:
c    c) ipi(1), ipi(1)+1  when imi(1)= 2 or 3

         if ( (imi(1).eq.2) .or. (imi(1).eq.3) ) then

                 stop 'Case not modelled: imi(1)= 2 or 3'
         else
c    c1) ipi(1) when imi(1)= 1 or 4

          J=IPI(1)
         pdt = nie(2,j)   *exp(wp(2,j)   -wp(1,j))
         pdt1= nie(2,j+1)*exp(wp(2,j+1)-wp(1,j+1))
         idp= (par(18,j)   * (pdt-pdt0(j))    +
     +         par(18,j+1)* (pdt1-pdt0(j+1)) )/dt

         B(J)=B(J)  - idp
         S(J)=S(J)  + par(18,j)* pdt/dt
         S(IPU+J)=S(IPU+J) + par(18,j+1)* pdt1/dt

c    c2) ipi(1)+1 when imi(1)= 1 or 4

c         J=IPI(1)+1 --> no correction is needed

         endif

       RETURN
       END



c************************ subroutine RPIG ****************************
c-----------------------------------------------------------------------
c
       SUBROUTINE RPIG(I,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
c
      implicit none
c

      INTEGER
     + I,M,NBMAX

      DOUBLE PRECISION
     + RAB,RS,DRAB,DRS,WP(NBMAX,M),WI(1,M),
     + PAR(19,0:M+1),NIE(2,M)

      DOUBLE PRECISION
     + FERM12,N0,P0,NI,PI,NTI,PTI,R,D,RAE,RAH

      EXTERNAL FERM12

C calculation of shockley-read-hall (rs) and auger plus direct
C band to band (rab) recombination rates at i grid point

      N0=PAR(13,I)*FERM12(PAR(16,I)+WI(1,I))/EXP(PAR(16,I))
      P0=PAR(14,I)*FERM12(PAR(17,I)-WI(1,I))/EXP(PAR(17,I))

      NI=NIE(1,I)*EXP(WP(1,I)-WP(3,I))
      PI=NIE(2,I)*EXP(WP(2,I)-WP(1,I))
      NTI=SQRT(N0*P0)*EXP(PAR(7,I))
      PTI=SQRT(N0*P0)*EXP(-PAR(7,I))



      R=NI*PI-N0*P0
      D=PAR(5,I)*(PI+PTI)+PAR(6,I)*(NI+NTI)

      RS=R/D

      RAE=(NI*NI)*PI
      RAH=NI*(PI*PI)
      RAB=R*PAR(10,I)+PAR(8,I)*(RAE-(N0*N0)*P0)+
     +                PAR(9,I)*(RAH-N0*(P0*P0))


c calculation of derivatives with respect to quasi-Fermi potential
c  for holes of each recombination rate

      DRS =(NI*PI*D-R*PAR(5,I)*PI)/(D*D)
      DRAB= PAR(10,I)*NI*PI+PAR(8,I)*RAE+PAR(9,I)*2*RAH

      RETURN
      END



c************************ subroutine RPII ****************************
c-----------------------------------------------------------------------
c
```

```
      SUBROUTINE RPII(I,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX,
     +                  RI,DRI,IPI,PARINT)
c
      implicit none
c
      INTEGER
     + I,M,NBMAX,IPI(2)

      DOUBLE PRECISION
     + RAB,RS,DRAB,DRS,WP(NBMAX,M),WI(1,M),
     + PAR(19,0:M+1),PARINT(5,2),NIE(2,M),
     + RI,DRI(2)

      DOUBLE PRECISION
     + FERM12,N0,P0,NI,PI,NTI,PTI,R,D,RAE,RAH,
     + SI,PCA,PCP,N0A,P0A,NIA,PIA,NTIA,PTIA,DIN,NUM,DINA,NUMA

      EXTERNAL FERM12

C calculation of shockley-read-hall (rs) and auger plus direct
C band to band (rab) recombination rates at i grid point

      N0=PAR(13,I)*FERM12(PAR(16,I)+WI(1,I))/EXP(PAR(16,I))
      P0=PAR(14,I)*FERM12(PAR(17,I)-WI(1,I))/EXP(PAR(17,I))

      NI=NIE(1,I)*EXP(WP(1,I)-WP(3,I))
      PI=NIE(2,I)*EXP(WP(2,I)-WP(1,I))
      NTI=SQRT(N0*P0)*EXP(PAR(7,I))
      PTI=SQRT(N0*P0)*EXP(-PAR(7,I))



      R=NI*PI-N0*P0
      D=PAR(5,I)*(PI+PTI)+PAR(6,I)*(NI+NTI)

      RS=R/D

      RAE=(NI*NI)*PI
      RAH=NI*(PI*PI)
      RAB=R*PAR(10,I)+PAR(8,I)*(RAE-(N0*N0)*P0)+
     +                PAR(9,I)*(RAH-N0*(P0*P0))


c calculation of derivatives with respect to quasi-Fermi potential
c  for holes of each recombination rate

      DRS =(NI*PI*D-R*PAR(5,I)*PI)/(D*D)
      DRAB= PAR(10,I)*NI*PI+PAR(8,I)*RAE+PAR(9,I)*2*RAH


C calculation of the interface recombination


      IF(I.EQ.IPI(1)+1) THEN
            SI=PARINT(1,1)
            PCA=PARINT(2,1)
            PCP=PARINT(3,1)
      ELSE
            SI=PARINT(1,2)
            PCA=PARINT(2,2)
            PCP=PARINT(3,2)
      END IF

      N0A=PAR(13,I-1)*FERM12(PAR(16,I-1)+WI(1,I-1))/EXP(PAR(16,I-1))
      P0A=PAR(14,I-1)*FERM12(PAR(17,I-1)-WI(1,I-1))/EXP(PAR(17,I-1))
      NIA=NIE(1,I-1)*EXP(WP(1,I-1)-WP(3,I-1))
      PIA=NIE(2,I-1)*EXP(WP(2,I-1)-WP(1,I-1))

      NTIA=SQRT(N0A*P0A)*EXP(PCA)
      PTIA=SQRT(N0A*P0A)*EXP(-PCA)
      NTI=SQRT(N0*P0)*EXP(PCP)
      PTI=SQRT(N0*P0)*EXP(-PCP)

      DINA=NIA*PIA-N0A*P0A
      NUMA=NIA+PIA+NTIA+PTIA

      DIN=NI*PI-N0*P0
      NUM=NI+PI+NTI+PTI

      RI=SI*((DINA/NUMA)+(DIN/NUM))

      DRI(1)=SI*(NI*PI*NUM-PI*DIN)/(NUM*NUM)
      DRI(2)=SI*(NIA*PIA*NUMA-PIA*DINA)/(NUMA*NUMA)


      RETURN
      END






c************************** subroutine BER *****************************
c-----------------------------------------------------------------------
c
      SUBROUTINE BER(X,BF)
c
      implicit none
c

        double precision lmg
        parameter (LMG=0.690775d+03)

        double precision x,bf,ex


c       (mp = 1.0e-300)
c      MG=1/MP
c      LMG=DLOG(MG)

        if ((x*x).le.0.01) then
                bf= 1. - x/2. + (x*x/12.)*(1.-x*x/60.)
        else if ((x .gt. 0.1) .and. (x .lt. lmg)) then
                ex=exp(-x)
                bf= x * ex / (1. - ex)
        else if ((x .lt. -0.1) .and. (x .gt. -lmg)) then
                bf= x / (exp(x)-1.)
        else if (x .ge. lmg) then
                bf= 0.0
        else
                bf= -x
        endif

        return
        end




c************************ subroutine densi ***************************
c-------------------------------------------------------------------
c
       subroutine densi (M,W,PAR,IPI,IBT,NBMAX,NMAX,NIE,WI,PARINT,NFjt,
     + IMI,PTT,FET,PTTC,JC,VBE,AREA,T,NI,S0,AS,RC,RB,RE,ABA,ACO)

        implicit none

      INTEGER M,IPI(2),IBT,IMI(2),NBMAX,NMAX,LS,J0,JF,JJ,KT,I,NFjt
      DOUBLE PRECISION XX(600),W(NBMAX,NMAX),JP(600),JN(600),
     + JT(600),PAR(19,0:NMAX+1),NIE(2,NMAX),J(5),
     + CP,XP,BP,EFPP,EFP,EPP,CN,XN,BN,EFNN,EFN,EPN,
     + PI,PIM,PIP,NP,NA,PTT,FET(5,4),JRBA(6),PTTC,VCE,JBB,
     + JC,VBE,AREA,T(3),NI,S0,AS,RC,RB,RE,ABA,ACO,TT(2),BETA,BETA2,
     + IR(600),IRCOL,IRTOTAL,WI(1,NMAX),PARINT(5,2),
     + JRS,VT,XNOR8,JNOR,Cnorm


        VT = 0.025879
        XNOR8 = DSQRT(8.85418D-14*VT/16.0218) * 1.0d+08
        JNOR=16.0218/(XNOR8* 1.0d-08)
        Cnorm=1.0d+20

c     ***********************************************************
c      calculo de las densidades de corriente en cada punto
c     ***********************************************************

       DO 80 LS=0,2

       IF(LS.EQ.0) THEN
         J0=1
         JF=ipi(1)-1
       END IF
       IF(LS.EQ.1) THEN
         J0=ipi(1)
         JF=ipi(2)-2
       END IF
       IF(LS.EQ.2) THEN
         J0=ipi(2)-1
         JF=M-3
       END IF

       DO 80 JJ=J0,JF
           KT=JJ+LS
           XX(JJ)= par(19,JJ+LS)

c          *******************************************************
c          densidad de corriente de huecos: assume DD model
c          *******************************************************

           CP=(PAR(4,KT+1)*NIE(2,KT+1)+PAR(4,KT)*NIE(2,KT))/
     +              (2.*(PAR(1,KT+1)-PAR(1,KT)))
           XP=W(1,KT+1)-W(1,KT)
                CALL BER(XP,BP)
           EFPP=DEXP(W(2,KT+1))
           EFP=DEXP(W(2,KT))
           EPP=DEXP(W(1,KT))
           JP(JJ)=-CP*(EFPP-EFP)*BP/EPP

80       CONTINUE

c          *******************************************************
c          si en colector/base Ev es abrupta, es decir,
c          IMI(1)=2 o 3, entonces el transporte es por emision
c          termionica para Jp(c+1+1/2)
c          *******************************************************

          IF( (IMI(1).EQ.2) .OR. (IMI(1).EQ.3) ) THEN
            KT=IPI(1)+1
            PI=NIE(2,KT)*DEXP(W(2,KT)-W(1,KT))
            PIM=NIE(2,KT-1)*DEXP(W(2,KT-1)-W(1,KT-1))
            JP(ipi(1))=FET(3,1)*(PIM-PI*FET(4,1)*FET(5,1))
          END IF


c          *******************************************************
c          si en base/emisor Ev es abrupta, es decir,
c          IMI(2)=2 o 3, entonces el transporte es por emision
c          termionica para Jp(e-1/2)
c          *******************************************************

          IF( (IMI(2).EQ.2) .OR. (IMI(2).EQ.3) ) THEN
            KT=IPI(2)-1
            PIP=NIE(2,KT+2)*DEXP(W(2,KT+2)-W(1,KT+2))
            PI=NIE(2,KT+1)*DEXP(W(2,KT+1)-W(1,KT+1))
            JP(ipi(2)-2)=-FET(3,2)*(PIP-PI*FET(4,2)*FET(5,2))
          END IF


c  Alternativa per a calcular Jrecomb a tot el dispositiu:

          call recomb (M,IPI,IMI,IBT,
     +             T,W,WI,PAR,PARINT,NIE,NBMAX,IR,IRCOL,IRTOTAL)

c
c         ************************************************************
c         ************************************************************
c         colocar en los fichero 'DENHUE' y 'DENELE' las densidades de
c         corriente de huecos, electrones y total en cada punto
c         'denhue':x,Jt,Jp    'denele':x,Jt,Jn
cc
cc        ARA: fitxer 'MCoutfjt': x,Jp,IR,p,n
cc
c         ************************************************************
         DO 120 I=1,M-3
          JP(I)=JP(I)*JNOR
       WRITE(NFjt,69) XX(I)*XNOR8, JP(I),par(1,i)*xnor8, IR(I)*JNOR,
     +         NIE(2,I)*EXP(W(2,I)-W(1,I))*Cnorm,
     +         NIE(1,I)*EXP(W(1,I)-W(3,I))*Cnorm

120   CONTINUE
       WRITE(NFjt,70) ''

69     FORMAT(T1,F9.1,1(1X,G12.5),1X,F9.1,3(1X,G12.5))
70     FORMAT(A)

c      *****************************************************************
c      *****************************************************************
c      calculo de los potenciales y las corrientes en cada contacto,asi
c      como el valor de la ganancia de corriente en emisor comun y de
c      las corrientes de recombinacion:
c      J(1) es JC=(JT(1)+JT(2)+JT(3))/3
c      J(2) es JB=JT(ibt-1)-JT(ibt-2) sin JRS
c      J(3) es JE=(JT(M-3)+JT(M-4)+JT(M-5))/3
c      J(4) es corr. de huecos en el punto anterior al contecto de base
c      J(5) es corr. de huecos que parten del contacto de base al emisor
c      ****
c      JRBA(1) recom base parte de emisor =JP(ibt-1)-JP(IPI(2)-2)
c      JRBA(2) recom interfaz B/E =JP(IPI(2)-2)-JP(IPI(2)-1)
c      JRBA(3) recom emisor =JP(IPI(2)-1)-JP(M-3)
c      JRBA(4) recom base parte de colector =JP(IPI(1))-JP(ibt-2)
c      JRBA(5) recom interfaz base-colector =JP(IPI(1)-1)-JP(IPI(1))
c      JRBA(6) recom colector =JP(1)-JP(IPI(1)-1)
c      *****************************************************************
c      *****************************************************************

c      **********************************************************
c      calculo de las densidades de corriente en los terminales
c      **********************************************************

       J(2)=JP(ibt-1)-JP(ibt-2)
       J(4)=JP(ibt-2)
       J(5)=JP(ibt-1)

c Corrent d'electrons calculat per Monte Carlo
       J(1)=JC
       J(3)=JC


c      ***************************************************************
c      corrientes de recombinacion en el dispositivo
c      ***************************************************************
       JRBA(1)=JP(ibt-1)-JP(ipi(2)-2)
       JRBA(2)=JP(ipi(2)-2)-JP(ipi(2)-1)
       JRBA(3)=JP(ipi(2)-1)-JP(M-3)
       JRBA(4)=JP(ipi(1))-JP(ibt-2)
       JRBA(5)=JP(ipi(1)-1)-JP(ipi(1))
       JRBA(6)=JP(1)-JP(ipi(1)-1)



       TT(1)=(T(2)-T(3))*VT
       TT(2)=(T(1)-T(3))*VT

c      *****************
c      corriente de recombinacion superficial de SOL-1990
       JRS=((1.6028*1D-19*NI*S0*AS)/(AREA*2.))*DEXP((T(2)-T(3))/2.)

c      *****************
c      la corriente de base sera la suma de la calculada y de la
c      de recombinacion superficial
c      JBB, JB SIN JRS
C      J(2), JB CON JRS
       JBB=J(2)
       J(2)=JRS+J(2)

c      ****************
c      la tension base-emisor real sera diferente si tenemos en cuenta
c      la caida en RE y RC
       VBE=TT(1)+(ABA*RB*J(2)+AREA*RE*J(3))*1D-8

c      ***************
c      la caida de tension en colector y emisor tmambien cambia
c      debido a la caida en RC y RE
       VCE=TT(2)+(ACO*RC*J(1)+AREA*RE*J(3))*1D-8



c      ***************
c      la ganancia de corriente SIN JRS es BETA2
c      la ganancia de corriente CON JRS es BETA
       BETA2=J(1)/JBB
       BETA=J(1)/J(2)

c      ***************
c      recom.col: VBE, Jp(c),Jrc,Jribc,Jrbc,Jp(b-)
c      recom.emi: VBE, Jp(b+),Jrbe,Jribe,Jre,Jp(e)
c      carent: VBE,Jb(con Jrs),Jc,Jrs,Jb(sin Jrs)
c      carsal: VCE,Jc,Bf(con Jrs),Bf(sin Jrs)
c      ***************
       write(*,*)
       write(*,*)

       write(*,312)'recomb.col: VBE, Jp(c), Jrc, Jribc, Jrbc, Jp(b-)'
       WRITE(*,138) VBE,JP(1),JRBA(6),JRBA(5),JRBA(4),J(4)
       write(*,*)

       write(*,312)'recomb.emi: VBE, Jp(b+), Jrbe, Jribe, Jre, Jp(e)'
       WRITE(*,138) VBE,J(5),JRBA(1),JRBA(2),JRBA(3),JP(M-3)
       write(*,*)

       write(*,312)'carent: VBE, Jb(con Jrs), Jc, Jrs, Jb(sin Jrs)'
       WRITE(*,140) VBE,J(2),J(1),JRS,JBB
       write(*,*)

       write(*,312)'carsal: VCE, Jc, Bf(con Jrs), Bf(sin Jrs)'
       WRITE(*,150) VCE,J(1),BETA,BETA2
       write(*,*)
       write(*,*)


       write(*,*)'Recombinacio entre terminal C i B :   ',
     +                IRCOL*JNOR
       write(*,*)'Recombinacio entre terminal E i B :   ',
     +                (IRTOTAL-IRCOL)*JNOR
       write(*,*)'Recombinacio total = corrent base :   ',
     +                IRTOTAL*JNOR
       write(*,*)
```

```
        write(*,*)

c       ****FORMAT*********
138     FORMAT(1X,G12.6,1X,G12.6,1X,G12.6,1X,G12.6,1X,G12.6,1X,G12.6)
140     FORMAT(1X,G12.6,1X,G12.6,1X,G12.6,1X,G12.6,1X,G12.6)
150     FORMAT(1X,G12.6,1X,G12.6,1X,F14.2,1X,F14.2)
312     FORMAT(A)


       RETURN
       END





c************************ subroutine recomb *************************
c-------------------------------------------------------------------
c
        subroutine recomb (M,IPI,IMI,IBT,
     +  T,WP,WI,PAR,PARINT,NIE,NBMAX,IR,IRCOL,IRTOTAL)
c
      implicit none
c

      INTEGER
     + M,IPI(2),IMI(2),IBT,NBMAX

      DOUBLE PRECISION
     + T(3),WP(NBMAX,M),WI(1,M),PAR(19,0:M+1),PARINT(5,2),
     + NIE(2,M),IRCOL,IRTOTAL

      INTEGER
     + J

      DOUBLE PRECISION
     + RAB,RS,DRAB,DRS,IR(M),
     + DRI(2),RI

c  General case:

      DO J=2,IPI(1)-1
         CALL RPIG(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
         IR(J) =PAR(18,J)*(RAB+RS)
      ENDDO

      DO J=IPI(1)+2,IPI(2)-1
         CALL RPIG(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
         IR(J) =PAR(18,J)*(RAB+RS)
      ENDDO


      DO J=IPI(2)+2,M-1
         CALL RPIG(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
         IR(J) =PAR(18,J)*(RAB+RS)
      ENDDO

c  Corrections to the general case:
c   a)  1,M

        J=1
        CALL RPIG(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
        IR(J) =0.5*PAR(1,J+1)*(RAB+RS)

        J=M
        CALL RPIG(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
        IR(J) =0.5*(PAR(1,J)-PAR(1,J-1))*(RAB+RS)


c  Corrections to the general case:
c   b) ipi(2), ipi(2)+1

        if ( (imi(2).eq.2) .or. (imi(2).eq.3) ) then
c  when imi(2)= 2 or 3

         J=IPI(2)
        CALL RPIG(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
        IR(J) =PAR(18,J)*(RAB+RS)

         J=IPI(2)+1
        CALL RPII(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX,
     +                RI,DRI,IPI,PARINT)
        IR(J) =PAR(18,J)*(RAB+RS)+RI

        else
c  when imi(2)= 1 or 4

         J=IPI(2)
        CALL RPIG(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
        IR(J) =PAR(18,J)*(RAB+RS)

         J=IPI(2)+1
        CALL RPII(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX,
     +                RI,DRI,IPI,PARINT)
        IR(J) =PAR(18,J)*(RAB+RS)+RI

        endif


c  Corrections to the general case:
c   c) ipi(1), ipi(1)+1  when imi(1)= 2 or 3

        if ( (imi(1).eq.2) .or. (imi(1).eq.3) ) then

                stop 'Case not modelled: imi(1)= 2 or 3'
        else
c  when imi(1)= 1 or 4

         J=IPI(1)
        CALL RPIG(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX)
        IR(J) =PAR(18,J)*(RAB+RS)

         J=IPI(1)+1
        CALL RPII(J,RAB,RS,DRAB,DRS,WP,WI,PAR,NIE,M,NBMAX,
     +                RI,DRI,IPI,PARINT)
        IR(J) =PAR(18,J)*(RAB+RS)+RI

        endif
```

```
c Total recombination through the device

        IRTOTAL=0.
        do j=1,ibt-1
          IRTOTAL = IRTOTAL + IR(j)
        enddo

        IRCOL=IRTOTAL


        do j=ibt,M
          IRTOTAL = IRTOTAL + IR(j)
        enddo


      return
      end





c************************ subroutine frequen ************************
c--------------------------------------------------------------------
c
        subroutine frequen(PAR,NIE,W,IPI,M,FREPE,NMAX,NBMAX,
     +      TIC,AREA,IBT,XCELE,XCHUE,JCA,JC)


        INTEGER
     +     NMAX,NBMAX,IPI(2),M,FREPE,IBE,IBC,IBT

        DOUBLE PRECISION
     + PAR(19,0:NMAX+1),NIE(2,NMAX),W(NBMAX,NMAX),
     + XCELE(M),XCHUE(M),QT,
     + QE,QB,QCO,QBE,QBC,DTT,DTTS,DTB,DTE,DTC,DTBE,DTBC,
     + JP(3),JN(3),JT(3),IJC,
     + CT,GM,FTRA,JC,JCA,
     + CP,XP,BP,EFPP,EFP,EPP,CN,XN,BN,EFNN,EFN,EPN,
     + VT,XNOR8,JNOR,TIC,AREA



c       ***********************************************************
c       calculo de las constantes utilizadas
c       ***********************************************************
       VT = 0.025879
       XNOR8= DSQRT(8.85418D-14*VT/16.0218)*1.0d+08
       JNOR=16.0218/XNOR8*1.0d-08
       QNOR=XNOR8*1.0d-08*1.60128*1D-19


c       ***********************************************************
C       ++++++++++++++++++++
c       +++ FREPE=0 +++++++++
c       ***********************************************************
c       si frepe=0 guardamos en frecu.dat los valores de las n y p
c       ***********************************************************

       IF(FREPE.EQ.0) THEN

c       calculo de las concentraciones de electrones y huecos
c       a lo largo de todo el dispositivo

        DO 100 I=1,M
         XCHUE(I)=1.d+20*NIE(2,I)*DEXP(W(2,I)-W(1,I))
         XCELE(I)=1.d+20*NIE(1,I)*DEXP(W(1,I)-W(3,I))
100     CONTINUE

c       ***********************************************************
C       ++++++++++++++++++++
c       +++ FREPE=1 +++++++++
C       ++++++++++++++++++++
c       ***********************************************************
C       si es frepe=1 recuperamos del fichero
c       las cargas y la corriente de colector
c       ***********************************************************

        ELSE IF(FREPE.EQ.1) THEN


c       *************************************************
c       incremento de electrones y huecos en cada punto
c       *************************************************

        DO 130 I=1,M
         XCHUE(I)=1.d+20*NIE(2,I)*DEXP(W(2,I)-W(1,I))-XCHUE(I)
         XCELE(I)=1.d+20*NIE(1,I)*DEXP(W(1,I)-W(3,I))-XCELE(I)
130     CONTINUE

c       ***********************************
c       calculo del limite de cada region,
c       (0,xbc), tal que dn>dp,
c       xbc, es dn=dp
c       (xbe,1), tal que dn>dp,
c       xbe, es dn=dp
c       posicion de los limites de cada region para calcular
c       los tiempos de transito y de carga
c       ***********************************

        IBC=1
        IBE=M

        DO 140 I=IBT,1,-1
         IF(ABS(XCHUE(I)).LE.ABS(XCELE(I))) THEN
          IBC=I
          GOTO 150
         END IF
140     CONTINUE

150     CONTINUE

        DO 160 I=IBT,M,+1
         IF(ABS(XCHUE(I)).LE.ABS(XCELE(I))) THEN
          IBE=I
          GOTO 170
         END IF
 160    CONTINUE

 170    CONTINUE

c       *****************************************************
c       calculo de los incrementos de corriente
c       *****************************************************
        IJC=JC-JCA

c       ******************************************************
c       ******************************************************
c       calculo de las integrales para la carga, en cada zona
c       en todo el dispositivo
c       ******************************************************
c       ******************************************************



c       **********************************************************
c       incremento de carga de electrones desde emisor a colector
c       **********************************************************

        QT=0.
        DO 205 I=1,M
          QT = QT + ABS(XCELE(I))*PAR(18,I)
 205    CONTINUE

        QT=QT*QNOR

c       **********************************************************
c       en colector, para tiempo de transito colector
c       **********************************************************

        QCO=0.
        DO 210 I=1,IBC
          QCO = QCO + ABS(XCHUE(I))*PAR(18,I)
 210    CONTINUE

        QCO=QCO*QNOR

c       ************************************************************
c       en colector, para tiempo de carga de la region de vaciamiento
c       de colector-base
c       ************************************************************

        QBC=0.
        DO 220 I=1,IBC
          QBC = QBC +(ABS(XCELE(I))-ABS(XCHUE(I))) * PAR(18,I)
 220    CONTINUE

        QBC=QBC*QNOR

c       ***********************************************************
c       en base, tiempo de transito base
c       ***********************************************************

        QB=0.
        DO 230 I=IBC+1,IBE-1
          QB = QB + ABS(XCELE(I))*PAR(18,I)
 230    CONTINUE

        QB=QB*QNOR

c       *****************************************************
c       en base-emisor, tiempo de carga
c       *****************************************************

        QBE=0.
        DO 240 I=IBE,M
          QBE = QBE +(ABS(XCELE(I))-ABS(XCHUE(I))) * PAR(18,I)
 240    CONTINUE

        QBE=QBE*QNOR

c       **********************************************************
c       en emisor tiempo de transito
c       **********************************************************

c       en colector, para tiempo de transito colector

        QE=0.
        DO 250 I=IBE,M
          QE = QE + ABS(XCHUE(I))*PAR(18,I)
 250    CONTINUE

        QE=QE*QNOR

c       *************************************
c       calculo de los tiempos de retardo
c       *************************************

        DTT=QT/IJC
        DTC=QCO/IJC
        DTBC=QBC/IJC
        DTB=QB/IJC
        DTBE=QBE/IJC
        DTE=QE/IJC

c       *****************************************************
c       suma de todos los tiempos de transito en cada region
c       *****************************************************

        DTTS=DTC+DTBC+DTB+DTBE+DTE

c       ******************************************************
c       calculo de las caracteristicas de regimen dinamico,
c       suponiendo valida la aproximacion quasi-estacionaria
c       ******************************************************

        GM=(IJC/TIC)*AREA*1.D-8
        CT=(QT/TIC)*AREA*1.D-8
        FTRA=IJC/(2.*3.141593*QT)

c       **********************************************************
c       escribimos en los ficheros los valores obtenidos
c       **********************************************************

        write(*,*)
        write(*,*)
        write(*,*)' Small signal parameters:'
        write(*,*)' Jc[A/cm2], gm[mA/V], CT[F], tEC[s] , ft[Hz], tEC[s]'
        WRITE(*,410) JCA,GM,CT,DTT,FTRA,DTTS
        write(*,*)
        write(*,*)' Jc[A/cm2], tC[s], tBC[s], tB[s], tBE[s], tE[s] '
        WRITE(*,410) JCA,DTC,DTBC,DTB,DTBE,DTE
        write(*,*)
        write(*,*)


310     FORMAT(1X,G12.6,1X,G12.6,1X,G12.6,1X,G12.6,1X,G12.6)
410     FORMAT(1X,G12.6,1X,G12.6,1X,G12.6,1X,G12.6,1X,G12.6,
     + 1X,G12.6)

        END IF
c       **********************************************************
c       ++++ END IF  FREPE=1++++++++++++++++++++++++++++++++++++
C       ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++



        RETURN
        END

ceeee+i**#**** ****#**** ****#**** ****#**** ****#**** ****#**** ****#72------80
```

```
ceeee+i**#**** ****#**** ****#**** ****#**** ****#**** ****#**** ****#72------80
c
c************************ subroutine Poisson ************************
c-------------------------------------------------------------------------
c
        SUBROUTINE Poisson (PSI0,PARINT,T,IPI,
     + M,W,PAR,NIE,S,B,C,MP,NBMAX,NMAX,
     + NTMAX,ITERM,LPROX,ERROR,CORRE,ITER,CORL,TIP)

c
      implicit none
c
      INTEGER
     + IPI(2),NBMAX,NMAX,NTMAX,ITERM,ITER,TIP,M

      DOUBLE PRECISION PSI0(2),T(3),MP,
     + W(NBMAX,NMAX),PAR(19,0:NMAX+1),S(NTMAX),
     + B(NMAX),PROX,PROXA,COR,CORA,C(NMAX),
     + NIE(2,NMAX),PARINT(5,2),CORRE,LPROX,ERROR,CORL,
     + FNC,FNV,FERM12

      INTEGER
     + J

c       inilializamos algunos parametros utilizados

      ITER=0
      ERROR=0
      CORRE=0
      TIP=0
      PROX= 1.0D+10
      COR = 1.0D+10

c       ********************************************************
c       ********************************************************
c       COMIENZA EL PROCESO ITERATIVO
C       *******************************************************

130     CONTINUE

        ITER=ITER+1

c       ****************************************
c       actualizamos los valores de proxa y cora
c       ****************************************
        PROXA=PROX
        CORA=COR

c       ***************************************
c       actualizamos los valores de nien y niep
c       ***************************************


        DO 2500 J=1,M
         FNC = (W(1,J)-W(3,J)) + PAR(16,J)
         FNV = (W(2,J)-W(1,J)) + PAR(17,J)
         NIE(1,J) = PAR(13,J)*FERM12(FNC)/EXP(FNC)
         NIE(2,J) = PAR(14,J)*FERM12(FNV)/EXP(FNV)
2500   CONTINUE

c       *****************************************
c       calculamos unos nuevos valores de S y B
c       *****************************************

        CALL SETMATp(M,PSI0,PARINT,T,IPI,S,B,
     +  W,PAR,NIE,NBMAX,NTMAX)

c       ****************************************
c       calculo de f1,f2,f3, son cercanas a cero
c       ****************************************

        CALL RHSAV(PROX,B,M)

c       ****************************************************
c       solucionamos el sistema de ecuaciones para hallar dx
c       ****************************************************

      CALL LINSOL(IPI,S,B,C,M,MP,NBMAX,NMAX,NTMAX)

c       ************************************
c       calculo de los errores en dx
c       ************************************

      CALL RHSAV(COR,C,M)


      DO 220 J=1,M
        W(1,J)=W(1,J)+C(J)
220     CONTINUE


c       *****************************************
c       *****************************************
c       si la nueva solucion es la exacta salimos
c       *****************************************

        IF(PROX.LT.LPROX.AND.COR.LT.CORL) THEN
                TIP=1
                ERROR=PROX
                CORRE=COR
                RETURN


c       ****************************************************
c       ****************************************************
c       si la nueva solucion mejora la anterior, permitimos
c       seguir iterando para mejorarla aun mas
c       ****************************************************

        ELSE IF((PROX.LT.PROXA.OR.PROX.LT.1D-10).
     +           AND.COR.LT.CORA) THEN
                GOTO 130


c       *****************************************************
c       *****************************************************
c       si el numero de iteraciones supera el maximo permitido
c       salimos indicando que no converge (TIP = 2)
```

```
c       *****************************************************

        ELSE IF(ITER.GT.ITERM) THEN
                TIP=2
                ERROR=PROX
                CORRE=COR
                RETURN
c       ****************************************************
c       ****************************************************
c       si la nueva solucion hace f1,f2,f3 aprox cero
c       pero dx crece, reducimos el valor de dx en aquellos
c       puntos que ha aumentado demasiado y calculamos los w ???
c       ****************************************************

        ELSE IF(COR.GE.CORA.OR.(PROX.GE.1D-10.AND.COR.LT.CORA)) THEN
                GOTO 130

        END IF

c       *************************************************
c       errores cometidos en la simulacion numerica
c       *************************************************
      TIP=1
      ERROR=PROX
      CORRE=COR
      RETURN
      END


c************************ function FERMI12 **************************
c--------------------------------------------------------------------
c
        DOUBLE PRECISION FUNCTION FERM12(G)

c
      implicit none
c

      DOUBLE PRECISION G,SQPI,A,B,C,D,E,F

      SQPI = SQRT(3.14159265d+00)

        F=ABS(G-2.13)**2.4
        E=(F+9.6)**(1.0/2.4)
        D=G+2.13
        C=(D+E)**1.5
        B=(3.*SQPI)/(2.**0.5)
        A=EXP(-G)

        FERM12 = 1.0 / (A+(B/C))

      END


c************************ function INVFER ***************************
c--------------------------------------------------------------------
c
        DOUBLE PRECISION FUNCTION INVFER(U)

c
      implicit none
c

        DOUBLE PRECISION U,SQPI,A,B,C,D,E

      SQPI = SQRT(3.14159265d+00)

        A=LOG(U)
        B=1.0-(U*U)
        C=((3.*SQPI*U)/4.)**(2.0/3.0)
        E=(0.24+1.08*C)**(-2)
        D=1.+E

        INVFER=(A/B)+(C/D)

        END


c************************ subroutine RHSAV **************************
c--------------------------------------------------------------------
c
      SUBROUTINE RHSAV(AV,B,N)

c
      implicit none
c

      INTEGER
     + N

      DOUBLE PRECISION AV,B(N)

      INTEGER
     + K

C "Norm or average" of vector B

      AV = 0.
      DO 10 K=1,N
            AV = AV + B(K)*B(K)
10    CONTINUE

      AV = SQRT(AV)/N

      RETURN
      END


c************************ subroutine SETMATp ************************
c--------------------------------------------------------------------
c
        SUBROUTINE SETMATp(M,PSI0,PARINT,T,IPI,S,B,
     +  WP,PAR,NIE,NBMAX,NTMAX)

c
```

```
      implicit none
c
      INTEGER
     + M,IPI(2),NBMAX,NTMAX

      DOUBLE PRECISION
     + S(NTMAX),B(M),
     + T(3),WP(NBMAX,M),PAR(19,0:M+1),PARINT(5,2),
     + PSI0(2),NIE(2,M)

      INTEGER
     + IPL,IPU,IPE,J

      DOUBLE PRECISION
     + PK,NK,PKP1,NKP1,PE,NE,EDHK,EDHKM1,EDHKP1

C        pointers for one-dimensional array s
      IPL=M-1
      IPU=2*M-1
      IPE=3*M-2


C        setting up s and b matrices

      DO 10 J=2,IPI(1)-1

         EDHK  =(PAR(15,J+1)+PAR(15,J))/(2.*(PAR(1,J+1)-PAR(1,J)))
         EDHKM1=(PAR(15,J)+PAR(15,J-1))/(2.*(PAR(1,J)-PAR(1,J-1)))
         PK = NIE(2,J)*EXP(WP(2,J) - WP(1,J))
         NK = NIE(1,J)*EXP(WP(1,J) - WP(3,J))

        B(J)=
     +  -EDHK   *(WP(1,J+1)-WP(1,J  ))    +
     +   EDHKM1*(WP(1,J  )-WP(1,J-1))    -
     +   PAR(18,J)*(PK-NK+PAR(2,J))

        S(J)=-EDHK-EDHKM1-PAR(18,J)*(PK+NK)
        S(IPL+J)=EDHKM1
        S(IPU+J)=EDHK

10    CONTINUE

      DO 20 J=IPI(1)+2,IPI(2)-1

         EDHK  =(PAR(15,J+1)+PAR(15,J))/(2.*(PAR(1,J+1)-PAR(1,J)))
         EDHKM1=(PAR(15,J)+PAR(15,J-1))/(2.*(PAR(1,J)-PAR(1,J-1)))
         PK = NIE(2,J)*EXP(WP(2,J) - WP(1,J))
         NK = NIE(1,J)*EXP(WP(1,J) - WP(3,J))

        B(J)=
     +  -EDHK   *(WP(1,J+1)-WP(1,J  ))    +
     +   EDHKM1*(WP(1,J  )-WP(1,J-1))    -
     +   PAR(18,J)*(PK-NK+PAR(2,J))

        S(J)=-EDHK-EDHKM1-PAR(18,J)*(PK+NK)
        S(IPL+J)=EDHKM1
        S(IPU+J)=EDHK

20    CONTINUE

      DO 30 J=IPI(2)+2,M-1

         EDHK  =(PAR(15,J+1)+PAR(15,J))/(2.*(PAR(1,J+1)-PAR(1,J)))
         EDHKM1=(PAR(15,J)+PAR(15,J-1))/(2.*(PAR(1,J)-PAR(1,J-1)))
         PK = NIE(2,J)*EXP(WP(2,J) - WP(1,J))
         NK = NIE(1,J)*EXP(WP(1,J) - WP(3,J))

        B(J)=
     +  -EDHK   *(WP(1,J+1)-WP(1,J  ))    +
     +   EDHKM1*(WP(1,J  )-WP(1,J-1))    -
     +   PAR(18,J)*(PK-NK+PAR(2,J))

        S(J)=-EDHK-EDHKM1-PAR(18,J)*(PK+NK)
        S(IPL+J)=EDHKM1
        S(IPU+J)=EDHK

30    CONTINUE

c  Corrections to the general case:
c   a)  1,M; 2,M-1
c   a1) 1,M

        J=1
        B(J) = PSI0(1)+T(1)-WP(1,1)
        S(J) = 1.
        S(IPU+J) = 0.

        J=M
        B(J) = PSI0(2)+T(3)-WP(1,M)
        S(J) = 1.
        S(IPL+J) = 0.


c   a2) 2,M-1

        J=2
        PE = NIE(2,1)*EXP(WP(2,1)-WP(1,1))
        NE = NIE(1,1)*EXP(WP(1,1)-WP(3,1))
        B(J)=B(J)-0.5*PAR(1,2)*(PE-NE+PAR(2,1))
        S(IPL+J)=S(IPL+J)-0.5*PAR(1,2)*(PE+NE)


        J=M-1
        PE = NIE(2,M)*EXP(WP(2,M)-WP(1,M))
        NE = NIE(1,M)*EXP(WP(1,M)-WP(3,M))
        B(J)=B(J)-0.5*(PAR(1,M)-PAR(1,M-1))*(PE-NE+PAR(2,M))
        S(IPU+J)=S(IPU+J)-0.5*(PAR(1,M)-PAR(1,M-1))*(PE+NE)


c  Corrections to the general case:
c   b) ipi(2), ipi(2)+1

c   b1) ipi(2)
         J=IPI(2)
         EDHKM1 =
     +     (PAR(15,J)+PAR(15,J-1))  /(2.*(PAR(1,J)  -PAR(1,J-1)))
         EDHKP1 =
     +     (PAR(15,J+2)+PAR(15,J+1))/(2.*(PAR(1,J+2)-PAR(1,J+1)))
         PK    = NIE(2,J  )*EXP(WP(2,J  )-WP(1,J  ))
         NK   = NIE(1,J  )*EXP(WP(1,J  )-WP(3,J  ))
         PKP1 = NIE(2,J+1)*EXP(WP(2,J+1)-WP(1,J+1))
         NKP1 = NIE(1,J+1)*EXP(WP(1,J+1)-WP(3,J+1))

        B(J)=
     +          -EDHKP1 *(WP(1,J+2)-WP(1,J+1))      +
     +           EDHKM1 *(WP(1,J)  -WP(1,J-1))      -
     +           PAR(18,J  )*(PK-NK+PAR(2,J))       -
     +           PAR(18,J+1)*(PKP1-NKP1+PAR(2,J+1)) -
     +           PARINT(4,2)

        S(J)    =-EDHKM1-PAR(18,J  )*(PK+NK)
        S(IPL+J)= EDHKM1
        S(IPU+J)=-EDHKP1-PAR(18,J+1)*(PKP1+NKP1)
        S(IPE+4)= EDHKP1

c   b2) ipi(2)+1
         J=IPI(2)+1

        B(J)=WP(1,J-1)-WP(1,J)
        S(J)=+1.0
        S(IPL+J)=-1.0
        S(IPU+J)=0.
        S(IPE+3)=0.

c  Corrections to the general case:
c   c) ipi(1), ipi(1)+1

c   c1) ipi(1)
         J=IPI(1)
         EDHKM1 =
     +     (PAR(15,J)+PAR(15,J-1))  /(2.*(PAR(1,J)  -PAR(1,J-1)))
         EDHKP1 =
     +     (PAR(15,J+2)+PAR(15,J+1))/(2.*(PAR(1,J+2)-PAR(1,J+1)))
         PK   = NIE(2,J  )*EXP(WP(2,J  )-WP(1,J  ))
         NK   = NIE(1,J  )*EXP(WP(1,J  )-WP(3,J  ))
         PKP1 = NIE(2,J+1)*EXP(WP(2,J+1)-WP(1,J+1))
         NKP1 = NIE(1,J+1)*EXP(WP(1,J+1)-WP(3,J+1))

        B(J)=
     +          -EDHKP1 *(WP(1,J+2)-WP(1,J+1))      +
     +           EDHKM1 *(WP(1,J)  -WP(1,J-1))      -
     +           PAR(18,J  )*(PK-NK+PAR(2,J))       -
     +           PAR(18,J+1)*(PKP1-NKP1+PAR(2,J+1)) -
     +           PARINT(4,1)

        S(J)    =-EDHKM1-PAR(18,J  )*(PK+NK)
        S(IPL+J)= EDHKM1
        S(IPU+J)=-EDHKP1-PAR(18,J+1)*(PKP1+NKP1)
        S(IPE+2)= EDHKP1

c   c2) ipi(1)+1
         J=IPI(1)+1

        B(J)=WP(1,J-1)-WP(1,J)
        S(J)=+1.0
        S(IPL+J)=-1.0
        S(IPU+J)=0.
        S(IPE+1)=0.

      RETURN
```

```
      END

c************************ subroutine LINSOL *************************
c-------------------------------------------------------------------
c
      SUBROUTINE LINSOL(IPI,S,B,X,N,MP,NBMAX,NMAX,NTMAX)

c
      implicit none
c

      INTEGER
     + IPI(2),N,NBMAX,NMAX,NTMAX

      DOUBLE PRECISION
     + S(NTMAX),B(NMAX),MP,X(NMAX)

      INTEGER
     + K

C        factorization of matrix s. the results l,d,u are overwritten on s

      CALL FACTOR(S,N,IPI,MP,NBMAX,NTMAX)

C        solve the system l.(d.u.x) = b by forward and back sustitution

        CALL SOLFBS(S,B,N,IPI,NBMAX,NMAX,NTMAX)

      DO 10 K=1,N
            X(K)=B(K)
10    CONTINUE

      RETURN
      END


c************************ subroutine FACTOR *************************
c-------------------------------------------------------------------
c
      SUBROUTINE FACTOR(S,N,IPI,MP,NBMAX,NTMAX)

c
      implicit none
c

      INTEGER
     + N,IPI(2),NBMAX,NTMAX

      DOUBLE PRECISION
     + MP,S(NTMAX)

      INTEGER
     + IPL,IPU,ICASE,I
      DOUBLE PRECISION
     + L,D,U,
     + ID,PROD,Z,E

C        define pointers for lower and upper codiagonals of matrix s

      IPL = N
      IPU = 2*N-1

C        initialization

            PROD=0
C        calculation of d,l,u. overwriting l,u and inverse of d on s.

      DO 20 I=1,N-1

C        for tridiagonal case

C        CALL SUBMS(I,S,PROD,D,NBMAX,NTMAX)
         D=S(I)-PROD

C        CALL COPM(D,ID)
         ID = D

C        CALL INVM(ID,MP,P,SF)
        IF (DABS(ID).LT.MP) THEN
          WRITE  (*,*) 'Zero diagonal. Floating DIVIDE trapped'
          ID=MP
        ENDIF
        ID=1.0/ID


c         modifications for "quasi-tridiag." case
         ICASE = 0
C              calculate l11 and/or l12, and overwrite on s
         IF(I.EQ.(IPI(1)-1)) ICASE=1
           IF(I.EQ.(IPI(2)-1)) ICASE=4
C              modificate lc and/or le, and calculate uu1 and/or uu2. owr.on s
           IF(I.EQ.IPI(1)) ICASE=2
           IF(I.EQ.IPI(2)) ICASE=5
C              modificate uc+1 and/or ue+1
           IF(I.EQ.(IPI(1)+1)) ICASE=3
           IF(I.EQ.(IPI(2)+1)) ICASE=6
           CALL MODTF(I,ID,L,U,S,N,ICASE,IPI,D,E,PROD,Z,NBMAX,NTMAX)



C           end of modifications for "quasi-tridiagonal" case

C        CALL MUL3M(L,D,U,E,PROD)
         E=D*U
         PROD=E*L

C        CALL OVERWS(I,S,ID,NBMAX,NTMAX)
         S(I) = ID

C        CALL OVERWS(IPL+I,S,L,NBMAX,NTMAX)
         S(IPL+I) = L

C        CALL OVERWS(IPU+I,S,U,NBMAX,NTMAX)
         S(IPU+I) = U

20    CONTINUE

C         calculation of last d, and overwrite its inverse on s

C      CALL SUBMS(N,S,PROD,D,NBMAX,NTMAX)
       D=S(N)-PROD

C       CALL COPM(D,ID)
        ID = D

C       CALL INVM(ID,MP,P,SF)
         IF (DABS(ID).LT.MP) THEN
           WRITE  (*,*) 'Zero diagonal. Floating DIVIDE trapped'
           ID=MP
         ENDIF
         ID=1.0/ID


C      CALL OVERWS(N,S,ID,NBMAX,NTMAX)
       S(N) = ID

       RETURN
       END


c************************ subroutine MODTF ***************************
c--------------------------------------------------------------------
c
       SUBROUTINE MODTF(I,ID,L,U,S,N,ICASE,IPI,D,E,PROD,Z,NBMAX,NTMAX)

c
       implicit none
c

       INTEGER
      + I,ICASE,NBMAX,NTMAX,N,IPE,INT,IPL,IPU,IPI(2)

       DOUBLE PRECISION
      + S(NTMAX),ID,D,E,
      + U, L,PROD,Z

C         define pointers for matrices l and u in vector s

       IPL = N
       IPU = 2*N - 1
       INT = IPI(1)
       IPE = 3*N - 2



C         calculate extra element (l11/l12) of matrix l. it is called e
C         overwrite e on s


       IF(ICASE.EQ.0) THEN

C        CALL MUL2MS(IPL+I,S,ID,L,NBMAX,NTMAX)
         L=ID*S(IPL+I)

C        CALL COPMS(IPU+I,S,E,NBMAX,NTMAX)
```

```
           E=S(IPU+I)

C          CALL MUL2M(ID,E,U)
           U=ID*E

         END IF


         IF(ICASE.EQ.1.OR.ICASE.EQ.4) THEN

            IF(ICASE.EQ.4) THEN
               IPE=IPE+2
            END IF

C           CALL MUL2MS(IPE+1,S,ID,E,NBMAX,NTMAX)
            E=ID*S(IPE+1)

C           CALL OVERWS(IPE+1,S,E,NBMAX,NTMAX)
            S(IPE+1) = E

C           CALL MUL2MS(IPL+I,S,ID,L,NBMAX,NTMAX)
            L=ID*S(IPL+I)

C           CALL COPMS(IPU+I,S,Z,NBMAX,NTMAX)
            Z=S(IPU+I)

C           CALL MUL2M(ID,Z,U)
            U=Z*ID

C           CALL MUL3M(E,D,U,PROD,Z)
            PROD=D*U
            Z=PROD*E

         END IF


C           calculate modification of old tridiag. element l in row c of s
C           and extra element of u, and overwrite it on s

         IF(ICASE.EQ.2.OR.ICASE.EQ.5) THEN

            IF(ICASE.EQ.5) THEN
               INT=IPI(2)
               IPE=IPE+2
            END IF

C           CALL COPMS(IPL+I,S,E,NBMAX,NTMAX)
            E=S(IPL+I)

C           CALL SUBM(E,Z,E)
            E=E-Z

C           CALL MUL2M(E,ID,L)
            L=ID*E

C           CALL COPMS(IPU+I,S,E,NBMAX,NTMAX)
            E=S(IPU+I)

C           CALL MUL2M(ID,E,U)
            U=ID*E

C           CALL COPMS(IPE+2,S,Z,NBMAX,NTMAX)
            Z=S(IPE+2)

C           CALL MUL2M(ID,Z,E)
            E=Z*ID

C           CALL OVERWS(IPE+2,S,E,NBMAX,NTMAX)
            S(IPE+2) = E

C           CALL MUL2M(L,D,PROD)
            PROD=L*D

C           CALL MUL2M(PROD,E,Z)
            Z=PROD*E

         END IF

C           calculate modification of old tridig. element u in row c+1 of s

         IF(ICASE.EQ.3.OR.ICASE.EQ.6) THEN

C           CALL MUL2MS(IPL+I,S,ID,L,NBMAX,NTMAX)
            L=ID*S(IPL+I)

C           CALL COPMS(IPU+I,S,E,NBMAX,NTMAX)
            E=S(IPU+I)

C           CALL SUBM(E,Z,E)
            E=E-Z

C           CALL MUL2M(ID,E,U)
            U=E*ID

         END IF

         RETURN
         END
```

```
c************************ subroutine SOLFBS *************************
c-------------------------------------------------------------------
c
      SUBROUTINE SOLFBS(S,B,N,IPI,NBMAX,NMAX,NTMAX)

c
      implicit none
c

      INTEGER
     + N,IPI(2),NBMAX,NMAX,NTMAX

      DOUBLE PRECISION
     + S(NTMAX),B(NMAX),VY,
     + VPROD,VZ,VE

      INTEGER
     + IPL,IPU,IPE,I,ICASE,K

C        set pointers for matrix s

      IPL = N
      IPU = 2*N-1
      IPE = 3*N-2


C        first solve l.y = b for y, divide y/d, and overwrite on d

C        initialize matrix prod

           VPROD=0

C        calculate y by forward sustitution


      DO 20 I=1,N

C        CALL SUBVB(I,B,VPROD,VY,NBMAX,NMAX)
         VY=B(I)-VPROD


C        CALL OVERWB(I,B,VY,NBMAX,NMAX)
         B(I) = VY


c        CALL MULMV(IPL+I,S,VY,VPROD,NBMAX,NTMAX)
         VPROD = S(IPL+I)*VY


C           modify calculations for "quasi-tridiag."  case
            ICASE = 0
            IF(I.EQ.IPI(1)) ICASE=1
            IF(I.EQ.IPI(2)) ICASE=2
            IF(ICASE.NE.0) CALL MODTS(1,ICASE,S,N,B,IPI,VZ,VE,VPROD,
     +                                                NBMAX,NMAX,NTMAX)

20    CONTINUE

C            multiply (inv. of d).y,and overwrite the result on y
      DO 50 I=1,N

C        CALL COPVB(I,B,VY,NBMAX,NMAX)
         VY = B(I)

c        CALL MULMV(I,S,VY,VZ,NBMAX,NTMAX)
         VZ = S(I)*VY

C        CALL OVERWB(I,B,VZ,NBMAX,NMAX)
         B(I) = VZ


50    CONTINUE


C          calculate x by back sustitution

C          initialization

            VPROD = 0
C        calculate x by back sustitution

      DO 40 K=1,N
         I = N-K+1

C        CALL SUBVB(I,B,VPROD,VZ,NBMAX,NMAX)
         VZ=B(I)-VPROD


C        CALL OVERWB(I,B,VZ,NBMAX,NMAX)
         B(I) = VZ


c        CALL MULMV(IPU+I-1,S,VZ,VPROD,NBMAX,NTMAX)
         VPROD = S(IPU+I-1)*VZ

            ICASE = 0
            IF(I.EQ.(IPI(1)+1)) ICASE=1
            IF(I.EQ.(IPI(2)+1)) ICASE=2
            IF(ICASE.NE.0) CALL MODTS(2,ICASE,S,N,B,IPI,VZ,VE,VPROD,
     +                                                NBMAX,NMAX,NTMAX)


40    CONTINUE
      RETURN
      END




c************************ subroutine MODTS **************************
c-------------------------------------------------------------------
```

```
c
       SUBROUTINE MODTS(IYX,ICASE,S,N,B,IPI,VZ,VE,VPROD,NBMAX,NMAX,
     +                                                    NTMAX)
c
       implicit none
c

       INTEGER
     + N,NBMAX,NMAX,NTMAX,ICASE,IYX,INT,IPE,KE,KYX,IPI(2)

       DOUBLE PRECISION
     + S(NTMAX),VZ,B(NMAX),
     + VE,VPROD

       INT=IPI(1)
       IPE=3*N-2

       IF(ICASE.EQ.2) THEN
          INT = IPI(2)
          IPE = IPE+2
       END IF


       KYX = -1
       KE = 1

       IF(IYX.EQ.2) THEN
          KYX = 2
          KE = 2
       END IF


c       CALL COPVB(INT+KYX,B,VZ,NBMAX,NMAX)
        VZ = B(INT+KYX)

c       CALL MULMV(IPE+KE,S,VZ,VE,NBMAX,NTMAX)
       VE = S(IPE+KE)*VZ

c      CALL ADD2V(VPROD,VE,VPROD)
       VPROD = VE+VPROD


       RETURN
       END


ceeee+i**#**** ****#**** ****#**** ****#**** ****#**** ****#**** ****#72------80
```

```
ceeee+i**#***** ****#***** ****#***** ****#***** ****#***** ****#***** ****#72------80
c
c************************ subroutine MCoutput ************************
c----------------------------------------------------------------------
c
c
c                   by Pau Garcias i Salva'. Nov., 1997
c
c----------------------------------------------------------------------
        subroutine MCoutput(tiphbt,matc,matb,mate,fmc,fmb,fme,tempK,M,
     +      ipi,ibt,imi,fet,t,psi0,efer,par,w,wi,nie,nbmax,nf)
c
c       storage allocation
c
c            implicit double precision (a-h,o-z)
c            implicit integer (i-n)

        implicit none
c
            integer
     +          tiphbt,matc,matb,mate,M,ipi(2),ibt,imi(2),nbmax
            double precision
     +          fmc,fmb,fme,tempK,t(3),psi0(2),efer,par(19,0:M+1),
     +          w(nbmax,M),wi(1,M),nie(2,M),fet(5,4)
            integer
     +          nf

            integer
     +          i
            logical
     +          zexist
            character*16
     +          fileout

c Comprovacio que la resolucio de Poisson recupera el potencial original
c  OK si dades dels fitxers DD2MC.dat i DD2MCres coincideixen, esp. w(i,j)

      fileout='MCoutf'
      INQUIRE (FILE= fileout,EXIST=zexist)
      if (.not.zexist) then
           OPEN (nf,STATUS='NEW',FORM='FORMATTED',FILE=fileout)
         else
           OPEN (nf,STATUS='OLD',FORM='FORMATTED',FILE=fileout)
      endif
      REWIND nf

        write(nf,90) tiphbt
        write(nf,90) matc,matb,mate
        write(nf,91) fmc,fmb,fme
        write(nf,90) M
        write(nf,90) ipi(1),ibt,ipi(2)
        write(nf,90) imi(1),imi(2)

        write(nf,*)
        write(nf,*)

        do 1000 i=1,M
          write(nf,90) i
          write(nf,91) par(1,i),par(2,i),par(3,i)
          write(nf,91) par(4,i),par(5,i),par(6,i)
          write(nf,91) par(7,i),par(8,i),par(9,i)
          write(nf,91) par(10,i),par(11,i),par(12,i)
          write(nf,91) par(13,i),par(14,i),par(15,i)
          write(nf,91) par(16,i),par(17,i)
          write(nf,*)
1000    continue

        write(nf,*)
        write(nf,*)


        do 1500 i=1,4
          write(nf,91) fet(1,i),fet(2,i),fet(3,i)
          write(nf,91) fet(4,i),fet(5,i)
1500    continue

        write(nf,*)
        write(nf,*)

        write(nf,91) t(1),t(2),t(3)
        write(nf,91) tempK
        write(nf,91) psi0(1),psi0(2),efer

        write(nf,*)
        write(nf,*)

        do 2000 i=1,M
          write(nf,90) i
          write(nf,91) w(1,i),w(2,i),w(3,i)
          write(nf,91) wi(1,i),nie(2,i),nie(1,i)
          write(nf,*)
2000    continue


      close (nf)
90    FORMAT (1X,3I5)
91    FORMAT (1X,3G23.16)

      return
      end

c************************ subroutine MCoutput2 ************************
c----------------------------------------------------------------------
c
c
c                   by Pau Garcias i Salva'. Jan., 1998
c
c----------------------------------------------------------------------
        subroutine MCoutput2(M,par,parNorm,Lnorm,w,gradp,nbmax,nf)

        implicit none

            integer
     +          M,LNorm,nbmax
            double precision
     +          par(19,0:M+1),parNorm(LNorm),w(nbmax,M),gradp(0:M+1)
            integer
     +          nf

            integer
     +          i
```

```
        double precision
     +        xnorm,Vtnorm,FnormKV


      Vtnorm = parNorm(3)
      xnorm  = parNorm(5)
      FnormKV   = - parNorm(9) * 1.0d-03


      do 20 i=1,M
        write(nf,92)
     +        par(1,i)*xnorm*1.0d+08,
     +        w(1,i)*Vtnorm,
     +        w(2,i)*Vtnorm,
     +        w(3,i)*Vtnorm,
     +        gradp(i)*FnormKV
20    continue

        write(nf,93) ''

92    FORMAT (T1,5(G14.7,2X))
93    FORMAT (A)

      return
      end

c************************* subroutine MCoutput3 ***********************
c----------------------------------------------------------------------
c
c
c                    by Pau Garcias i Salva'. Jan., 1998
c
c----------------------------------------------------------------------
      subroutine MCoutput3(M,par,parNorm,Lnorm,w,nie,npti,cn,nbmax,nf)

      implicit none

          integer
     +        M,LNorm,nbmax,npti(0:M+1)
          double precision
     +        par(19,0:M+1),parNorm(Lnorm),w(nbmax,M),nie(2,M),
     +              cn(0:M+1)
          integer
     +        nf

          integer
     +        i
          double precision
     +        cni,cpi,xnorm,Cnorm

      Cnorm  = parNorm(1)
      xnorm  = parNorm(5)


      do 20 i=1,M
        cpi=nie(2,i)*exp(w(2,i)-w(1,i))*Cnorm
        cni=nie(1,i)*exp(w(1,i)-w(3,i))*Cnorm
        write(nf,92)
     +        par(1,i)*xnorm*1.0d+08,
     +        cpi,
     +        cni,
     +        cn(i)*Cnorm,
     +        npti(i)
20    continue

        write(nf,93) ''

92    FORMAT (T1,4(G14.7,2X),I6)
93    FORMAT (A)

      return
      end

c************************* subroutine MCoutput4 ***********************
c----------------------------------------------------------------------
c
c
c                    by Pau Garcias i Salva'. Feb., 1998
c
c----------------------------------------------------------------------
      subroutine MCoutput4(par,parNorm,ipi,w,refp,pt,ipt,ec,cn,vxm,
     +        fd1,npt,time,alf,efm,epp,M,LNorm,nbmax,ivmax,iregnmax,
     +        nptmax,cn2,iwmax,wght,jdots,xlow,xhigh,nf,nf2,nf3)

      implicit none
c
          integer
     +        ikx,iky,ikz,its,ix,ival,icel,iwgt,ixm,idx,ixup
          parameter (ikx=1,iky=2,ikz=3,its=4,ix=5)
          parameter (ival=1,icel=2,iwgt=3,ixm=1,idx=18,ixup=19)
c
          integer
     +        ipi(2),ipt(3,nptmax),npt,nptmax,
     +        M,Lnorm,nbmax,ivmax,iregnmax,iwmax,jdots
          double precision
     +        par(19,0:M+1),parNorm(LNorm),w(nbmax,M),refp(5),
     +        cn(0:M+1),cn2(0:M+1),vxm(0:M+1),fd1(0:M+1),time,
     +        pt(5,nptmax),efm(ivmax,iregnmax),alf(ivmax,iregnmax),
     +        ec(ivmax,iregnmax),epp,wght(0:iwmax),xlow,xhigh
          integer
     +        nf,nf2,nf3

          integer
     +        iv,iregn,j,n,iw
          double precision
     +        x,kx,ky,kz,vx,ek,factorj,
     +        tnormps,xnorm,vnorm,Enorm,He,Hv,q,
     +        xj,xj1,cnj,alf2


      tnormps= parNorm(4)*1.0e+12
      xnorm= parNorm(5)
      vnorm= parNorm(6)
      Enorm= parNorm(7)

      He    = parNorm(20)
      Hv    = parNorm(22)

      q     = parNorm(31)


      do 1 j=0,M+1
```

```
          cn  (j)= 0.
          cn2 (j)= 0.
          vxm (j)= 0.
          fd1 (j)= 0.
 1      continue


        do 2 n=1,npt
          x    = pt(ix,n)
          kx   = pt(ikx,n)
          ky   = pt(iky,n)
          kz   = pt(ikz,n)
          j    = ipt(icel,n)
          iv   = ipt(ival,n)
          iw = ipt(iwgt,n)

          if ( j .gt. ipi(2)+1 ) then
                  iregn=3
          else if ( j .le. ipi(1) ) then
                  iregn=1
          else
                  iregn=2
          endif

          alf2 = 1.0  / (2.0*alf(iv,iregn))

          xj = par(ixm,j)
          xj1= par(ixm,j-1)
          cnj= (x-xj1)/(xj-xj1)

          cn(j-1) = cn(j-1) + (1.-cnj) *wght(iw)
          cn(j)   = cn(j)   + cnj      *wght(iw)

          cn2(j-1) = cn2(j-1) + (1.-cnj)
          cn2(j)   = cn2(j)   + cnj

          ek=(sqrt( 1.0+4.0*alf(iv,iregn)*He*(kx*kx+ky*ky+kz*kz)/
     +    efm(iv,iregn)) -1.0) * alf2

          vx=Hv*kx
     +    /(efm(iv,iregn)*( 1.0+2.0*alf(iv,iregn)*ek ))

          vxm (j)   = vxm (j)   + vx * cnj      *wght(iw)
          vxm (j-1) = vxm (j-1) + vx * (1.-cnj) *wght(iw)

          if (iv.eq.1) then
                  fd1(j)   = fd1(j)   + cnj      *wght(iw)
                  fd1(j-1) = fd1(j-1) + (1.-cnj) *wght(iw)
          endif


          if (jdots.eq.1) then
            if ( (x.gt.xlow) .and. (x.lt.xhigh) ) then
             if (iv.eq.1) then
              write(nf2,95) x*xnorm*1.0d+08, vx*vnorm,
     +            (-(w(1,j)*cnj+w(1,j-1)*(1.-cnj) +par(11,j))+
     +            ec(iv,iregn)+ek)*Enorm-refp(4),'G',iw
             else if (iv.eq.2) then
              write(nf2,95) x*xnorm*1.0d+08, vx*vnorm,
     +            (-(w(1,j)*cnj+w(1,j-1)*(1.-cnj) +par(11,j))+
     +            ec(iv,iregn)+ek)*Enorm-refp(4),'L',iw
             else
              write(nf2,95) x*xnorm*1.0d+08, vx*vnorm,
     +            (-(w(1,j)*cnj+w(1,j-1)*(1.-cnj) +par(11,j))+
     +            ec(iv,iregn)+ek)*Enorm-refp(4),'X',iw
             endif
            endif
          endif

 2        continue


 c  Correction related to the CIC method (half box integration; triangle shape):

 c        j=ipi(1)
 c        cn(j)=cn(j)+0.5*(cn(j)-cn(j-1)*par(idx,j)/par(idx,j-1))
 c        vxm(j)=vxm(j)+0.5*(vxm(j)-vxm(j-1))
 c        fd1(j)=fd1(j)+0.5*(fd1(j)-fd1(j-1))
 c        j=ipi(1)+1
 c        cn(j)=cn(j)-0.5*(cn(j+1)*par(idx,j)/par(idx,j+1)-cn(j))
 c        vxm(j)=vxm(j)-0.5*(vxm(j+1)-vxm(j))
 c        fd1(j)=fd1(j)-0.5*(fd1(j+1)-fd1(j))

 c !!! Only for homojunctions in the BC interface

          j=ipi(1)
          cn(j)=0.5*(cn(j)+cn(j+1)*par(idx,j)/par(idx,j+1))
          vxm(j)=0.5*(vxm(j)+vxm(j+1)*par(idx,j)/par(idx,j+1))
          fd1(j)=0.5*(fd1(j)+fd1(j+1)*par(idx,j)/par(idx,j+1))

          cn(j+1)=cn(j)*par(idx,j+1)/par(idx,j)
          vxm(j+1)=vxm(j)*par(idx,j+1)/par(idx,j)
          fd1(j+1)=fd1(j)*par(idx,j+1)/par(idx,j)

 c Valid for homo- & heterojunctions in the EB interface

          j=ipi(2)
          cn(j)=cn(j)+0.5*(cn(j)-cn(j-1)*par(idx,j)/par(idx,j-1))
          vxm(j)=vxm(j)+0.5*(vxm(j)-vxm(j-1)*par(idx,j)/par(idx,j-1))
          fd1(j)=fd1(j)+0.5*(fd1(j)-fd1(j-1)*par(idx,j)/par(idx,j-1))
          j=ipi(2)+1
          cn(j)=cn(j)-0.5*(cn(j+1)*par(idx,j)/par(idx,j+1)-cn(j))
          vxm(j)=vxm(j)-0.5*(vxm(j+1)*par(idx,j)/par(idx,j+1)-vxm(j))
          fd1(j)=fd1(j)-0.5*(fd1(j+1)*par(idx,j)/par(idx,j+1)-fd1(j))


          factorj = -q*epp*vnorm/(1.0d+04*xnorm)

          do 3 j=1,M
            vxm(j)=vxm(j)/cn(j)
            write(nf,92) par(ixm,j)*xnorm*1.0d+08, vxm(j)*vnorm,
     +            factorj*cn(j)*vxm(j)/par(idx,j),fd1(j)/cn(j)
 3        continue


          write(nf3,94) time*tnormps, npt

          write(nf2,93) ''
          write(nf ,93) ''

 92     FORMAT (T1,4(G14.7,2X),I6)
 93     FORMAT (A)
```

```
94    FORMAT (T1,1(G14.7,2X),I8)
95    FORMAT (T1,3(G14.7,2X),A,2X,I2)

      return
      end
c************************ subroutine Accumul ************************
c-------------------------------------------------------------------------
c
c
c                  by Pau Garcias i Salva'. Nov., 1997
c
c-------------------------------------------------------------------------
        subroutine Accumul (pt,ipt,npt,nptmax,epp,cn,vxm,ekm,fd1,
     +          Scn,Svx,Sek,Sfd1,cnaux,vxmaux,ekmaux,fd1aux,ipi,
     +          nmax,nthdsmax,nthds,LB,npti,par,M,efm,alf,gm,
     +          cn2,cn2aux,Scn2,iwmax,wght,parNorm,LNorm,ivmax,iregnmax)

c
c       storage allocation
c
c            implicit double precision (a-h,o-z)
c            implicit integer (i-n)

        implicit none
c
            integer
     +          ikx,iky,ikz,its,ix,ival,icel,iwgt,ixm,idx,ixup
            parameter (ikx=1,iky=2,ikz=3,its=4,ix=5)
            parameter (ival=1,icel=2,iwgt=3,ixm=1,idx=18,ixup=19)
c

            integer
     +          ipt(3,nptmax),npti(0:M+1),ipi(2),npt,nptmax,M,
     +          LNorm,ivmax,iregnmax,iwmax,
     +          nmax,nthdsmax,nthds,LB

            double precision
     +          pt(5,nptmax),
     +          cn(0:M+1),vxm(0:M+1),ekm(ivmax,0:nmax+1),fd1(0:M+1),
     +          Scn(M),Svx(M),Sek(ivmax,nmax),Sfd1(M),
     +          cnaux(0:nmax+1,nthdsmax),
     +          vxmaux(0:nmax+1,nthdsmax),
     +          ekmaux(ivmax,0:nmax+1,nthdsmax),
     +          fd1aux(0:nmax+1,nthdsmax),
     +          cn2(0:nmax+1),cn2aux(0:nmax+1,nthdsmax),Scn2(nmax),
     +          efm(ivmax,iregnmax),alf(ivmax,iregnmax),
     +          gm(ivmax,iregnmax),
     +          par(19,0:M+1),parNorm(LNorm),
     +          epp,wght(0:iwmax)


            integer
     +          n,j,jp,iv,iregn,k,kmax,ntd,iw
            double precision
     +          x,xj,xj1,cnj,He,Hv,ek,vx,alf2,kx,ky,kz


        He   = parNorm(20)
        Hv   = parNorm(22)

        kmax=npt/(nthds*LB)

        do 1 j=0,M+1
          cn(j)= 0.0
          cn2(j)= 0.0
          vxm(j)= 0.0
          ekm(1,j)= 0.0
          ekm(2,j)= 0.0
          fd1(j)= 0.0
1       continue

c$doacross local(k,j),
c$& shared(nthds,M,cnaux,cn2aux,vxmaux,ekmaux,fd1aux),
c$& mp_schedtype = simple
c$& , affinity(k) = data (cnaux(j,k))

        do 2 k=1,nthds
          do 3 j=0,M+1
            cnaux(j,k) = 0.0
            cn2aux(j,k) = 0.0
            vxmaux(j,k) = 0.0
            ekmaux(1,j,k) = 0.0
            ekmaux(2,j,k) = 0.0
            fd1aux(j,k) = 0.0
3         continue
2       continue


c$ doacross local(ntd,k,n,x,kx,ky,kz,jp,iv,iw,iregn,xj,xj1,cnj,
c$& ek,vx,alf2),
c$& shared(nthds,kmax,npt,LB,pt,ipt,par,ipi,cnaux,cn2aux,
c$& vxmaux,ekmaux,fd1aux,He,Hv,wght),
c$& mp_schedtype = simple
c$& ,affinity(ntd) = thread ( ntd-1 )
c..$& ,affinity(ntd) = data ( pt (1, 1+LB*(ntd-1+(k-1)*nthds)))
        do 110 ntd=1,nthds
        do 100 k=0,kmax
        do 120 n=1+LB*(ntd-1+k*nthds),
     +          min(npt, LB*(ntd+k*nthds))
          x   = pt(ix,n)
          kx   = pt(ikx,n)
          ky   = pt(iky,n)
          kz   = pt(ikz,n)
          jp   = ipt(icel,n)
          iv   = ipt(ival,n)
          iw   = ipt(iwgt,n)

          if ( jp .gt. ipi(2)+1 ) then
                  iregn=3
          else if ( jp .le. ipi(1) ) then
                  iregn=1
          else
                  iregn=2
          endif

          alf2 = 1.0  / (2.0*alf(iv,iregn))

          xj = par(ixm,jp)
          xj1= par(ixm,jp-1)
          cnj= (x-xj1)/(xj-xj1)

          cnaux(jp,ntd)   = cnaux(jp,ntd)   + cnj      *wght(iw)
        cnaux(jp-1,ntd) = cnaux(jp-1,ntd) + (1.-cnj) *wght(iw)

        cn2aux(jp,ntd)   = cn2aux(jp,ntd)   + cnj
        cn2aux(jp-1,ntd) = cn2aux(jp-1,ntd) + (1.-cnj)

        ek=(sqrt( 1.0+4.0*alf(iv,iregn)*He*(kx*kx+ky*ky+kz*kz)/
     +    efm(iv,iregn)) -1.0) * alf2

        ekmaux(iv,jp  ,ntd)=ekmaux(iv,jp  ,ntd) + ek*  cnj    *wght(iw)
        ekmaux(iv,jp-1,ntd)=ekmaux(iv,jp-1,ntd) + ek* (1.-cnj)*wght(iw)

        vx=Hv*kx
     +    /(efm(iv,iregn)*( 1.0+2.0*alf(iv,iregn)*ek ))

        vxmaux (jp,ntd)   = vxmaux (jp,ntd)   + vx * cnj      *wght(iw)
        vxmaux (jp-1,ntd) = vxmaux (jp-1,ntd) + vx * (1.-cnj) *wght(iw)


cc  ( a 2-valley model is programed and assumed )
        if (iv.eq.1) then
                fd1aux(jp,ntd)   = fd1aux(jp,ntd)   + cnj     *wght(iw)
                fd1aux(jp-1,ntd) = fd1aux(jp-1,ntd) + (1.-cnj)*wght(iw)
        endif

120   continue
100   continue
110   continue

      do 150 k=1,nthds
       do 160 j=1,M
         cn(j) = cn(j) + cnaux(j,k)
         cn2(j) = cn2(j) + cn2aux(j,k)
         vxm(j) = vxm(j) + vxmaux(j,k)
         ekm(1,j) = ekm(1,j) + ekmaux(1,j,k)
         ekm(2,j) = ekm(2,j) + ekmaux(2,j,k)
         fd1(j) = fd1(j) + fd1aux(j,k)
160   continue
150   continue

       do 200 j=1,M
         Scn(j)  = Scn(j)  +  cn(j)
         Scn2(j) = Scn2(j) +  cn2(j)
         Svx(j) = Svx(j) + vxm(j)
         Sek(1,j) = Sek(1,j) + ekm(1,j)
         Sek(2,j) = Sek(2,j) + ekm(2,j)
         Sfd1(j)=Sfd1(j) +  fd1(j)
200   continue

      return
      end

c-----------------------------------------------------------------------
ceeee+i**#**** ****#**** ****#**** ****#**** ****#**** ****#**** ****#72------80
c-----------------------------------------------------------------------
```