```
! --- MODIFIED TO INCLUDE NHC-EWALD-COUPLED POTENTIAL STUFF
! --- NMR refinement are removed
!
      PROGRAM SANDER
!
!        SANDER, version 4.1 (with NHC and EWALD)
!
! The Molecular Dynamics/NMR Refinement/Modeling Module of the AMBER Package.
!
!     This is a modified version of the AMBER 3.0, Rev. A MD Module which
!     includes an extensive suite of utilites for use with NMR refinement,
!     other modeling calculations, as well as other new options within
!     the bulk of the MD program (minimization is also included).
!
!     The NMR refinement/modeling suite, hooks thereto, and the new options
!     were written by
!
!             David A. Pearlman (UCSF)
!             David A. Case (Scripps) &
!             Ping Yip      (Scripps)
!
!     Version 4.1 also include the capability to carry out polarization
!     calculations. The polarizability code was written by Jim Caldwell
!     and Liem X. Dang (UCSF).  Truncated octahedral periodicity was
!     added by Thomas Huber of Ludwig Maximilian Universitaet Muenchen.
!
!     Additional changes for version 4.1 are listed below.
!
!     Revision A of version 3 of AMBER MD, on which this program was built,
!     was by George L. Seibel. Better vectorized nonbond routines
!     in version 3A were written by Rad Olson and Bill Swope (IBM).
!
!     Version 3 of AMBER MD was written by U.C. Singh and Peter A. Kollman,
!     adapting significantly from GROMOS83 by Wilfred van Gunsteren.
!
!*************************************************************************
!                             AMBER                                     **
!                                                                       **
! Copyright (c) 1986, 1991 Regents of the University of California      **
!                        All Rights Reserved.                           **
!                                                                       **
!  This software provided pursuant to a license agreement containing    **
!  restrictions on its disclosure, duplication, and use. This software **
!  contains confidential and proprietary information, and may not be    **
!  extracted or distributed, in whole or in part, for any purpose       **
!  whatsoever, without the express written permission of the authors.   **
!  This notice, and the associated author list, must be attached to     **
!  all copies, or extracts, of this software. Any additional            **
!  restrictions set forth in the license agreement also apply to this   **
!  software.                                                            **
!*************************************************************************
!
!  Significant changes for Version 4.1:
!
!     1) Inclusion of polarization code.
!        (J. Caldwell and L. Dang)
!     2) Use of fast analytical shake for 3 point waters.
!        (D. Pearlman and S. Miyamoto)
!     3) New much faster routines to handle TIP3P-TIP3P water interactions
!        (D. Case and D. Pearlman)
!     4) Inclusion of standard and time-averaged J-coupling restraints
!        (D. Pearlman)
!     5) New methods for NMR Intensity refinement and ring current calcs.
!        (D. Case)
!     6) Allow for dual non-bonded cutoffs.
!         (D. Pearlman)
!      7) New VLIMIT option to limit max. atomic velocity.
!         (D. Pearlman)
!      8) Incorporation of PEACS constant nrg contour conf. search capability
!         (D. Case)
!      9) Plus various minor modifications for clearer output, cleanup,
!         to incorperate bugfixes, etc.
!     10) New code for truncated octahedral periodic boundary conditions
!         (Thomas Huber, Ludwig Maximilian Universitaet Muenchen,
!          email: thuber@@Physik.TU-Muenchen.de) and reorganization/relocation
!          of imaging routines to period.f (B. Ross).
!
!  Changes for Version 4 (NMR):
!
!      Aside from the various hooks required to
!      integrate the NMR package,
!      1) the nonbonded code in nonbon and ephi
!         was modified to allow a "soft repulsion" non-bonded potential in
!         place of 6-12 or 10-12 vdw interactions;
!      2) several new temperature-coupling options were added in RUNMD.
!         These will be particularly useful when carrying out simulations
!         where the internal energy of the molecular system is changing
!         very quickly (such as in some MD/NMR refinement schemes).
!      3) The "NMR" package itself, which allows
!         a large number of simulation protocols appropriate for NMR/MD
!         refinements and general modeling work, and offers a relatively
!         flexible and easy-to-use interface. See the SANDER refinement
!         manual for more details.
!
! ============================================================================
!
      implicit double precision (a-h,o-z)

      LOGICAL SKIP, BELLY, erstop
#include "files.h"
#include "sizes.h"
#include "memory.h"
#include "box.h"
#include "md.h"
#include "parms.h"
#ifdef ROAR_CP
#include "cp.h"
#endif

#ifdef MPI
#include "mpif.h"
#include "parallel.h"
#endif

#include "nhc.h"
!
#include "ewald.h"
#include "pmedim.h"
#include "pme.h"
!
      dimension X(MAXREA)
      integer ix(MAXINT)
      integer ih(MAXHOL)
      dimension ene(30)
      real wtim0, wtim1

!
! Initialize the cpu timer. Needed for machines where returned cpu times
! are relative.
!
```

```
      CALL TIMIT(3,SKIP,6)

#ifdef HP
!
!   --- set up certain underflow operations
!
      on double precision underflow call trapud
      on real underflow call trapu
#endif
#ifdef AIX
        call setrteopts('namelist=old')
#endif
!
!     --- Code Configuration Section ---
!
!     Revision A code is designed to port easily to machines of
!     varying wordsize.   We have changed the memory mapping scheme
!     of version 3.0 to improve portabilty.  Instead of one large
!     array there are now three separate arrays.  The following points
!     are notable:
!     1)  Three arrays X, IH, and IX are passed as arguments instead
!         of COMMON.  These arrays are for Reals, Hollerith ints, and
!         Integers, respectively.  The Reals may be 32 or 64 bit, and
!         Integers may be 32 or 64 bit as well.   The Hollerith int
!         and numerical int arrays share the same starting address,
!         by virtue of the equivalence in main.  This is *not* non-portable,
!         don't get excited.  All of the structural arrays in the program
!         are mapped into the three large arrays, and are referenced
!         by the offsets passed in commons MEMLA for reals, MEMLB and
!         MEMLD for ints, and MEMLC for Hollerith data.
!     2)      Two variables are used to describe the packing of the
!         nonbonded pairlist.   IPACK is set to 1 if explicit word-
!         packing routines are called, as on Cray or FPS.  Otherwise
!         IPACK = 0.  NWDVAR is the number of NB pair pointers that is
!         held in a default integer word.   This will usually be 4 for
!         the Cray, 2 or 4 for the FPS, and on a 32 bit machine will
!         be 1 if the entire word is used, or 2 if an integer*2 declaration
!         is used for the pairlist.
!         NATIVE is the number of bits in a default integer on the target
!         machine.  It is usually 32 or 64.
!     3)      Memory Requirements:  The three parameters MAXREA, MAXINT,
!         and MAXDUP are used to control memory use.
!
!         Array         Use       Parameter      Typical Value
!           X       floating pt    MAXREA         ~ 23 * Natom
!          IX       Integers       MAXINT        ~ 150 * Natom
!         (various) dihedral dup   MAXDUP      0 - 1000, data dependent
!
!         The typical values given are only rough estimates.  The Integer
!         memory requirement consists of a "static" requirement, which
!         is topology dependent and does not vary throughout the run,
!         and a variable amount for the nonbonded pairlist pointers.
!         The value of ~150*Natom includes both the static requirement
!         and the pairlist requirement for a "typical" system assuming
!         a full word is used to store a pairlist pointer.   To determine
!         the actual Integer memory requirement, add the static requirement
!         reported at the start of a run to the pairlist requirement.  The
!         pairlist requirement is the total number of nonbonded pairs (this
!         is geometry and cutoff dependent) divided by NWDVAR.  If IPACK
!         is not 0, you should add Natom.  Since the pairlist can grow
!         during a run (and often does) it is a good idea to increase the
!         room for it by ~10%.   The maximum number of nonbonded pairs
!         for the value of MAXINT used will be reported at the top of the
!         output.
!
```

```
!         IPACK=0:   MAXINT = Static Int + (NPAIRS/NWDVAR)*1.1
!         IPACK=1:   MAXINT = Static Int + (NPAIRS/NWDVAR)*1.1+NATOM
!
!         All cases: MAXREA = reported static output.
!
!         Dihedrals that have more than one fourier term will have their
!         pointers duplicated for the vectorized dihedral routine.  This
!         is done twice; once for heavy atom dihedrals and once for diheds
!         involving H-atoms.   MAXDUP must be at least as large as the
!         larger number reported in the output.  The actual amount of space
!         allocated is 10*MAXDUP, so it should not be set too large in a
!         tight memory environment.
!
#ifdef MPI
!     Set up parallel execution
!
       CALL mpi_init(ierr)
       CALL mpi_comm_rank(MPI_COMM_WORLD,mytaskid,ierr)
       CALL mpi_comm_size(MPI_COMM_WORLD,numtasks,ierr)
!
!     Make PE 0 the master
      master = mytaskid.EQ.0
#endif
      NRU = 0
      NHCSETQ = .TRUE.
!
      erstop = .false.
!     --- generic packing scheme ---
      nwdvar = 1
      native = 32
      nlink =0
!
#ifdef ISTAR2
!     --- Int*2 packing scheme ---
      nwdvar = 2
#endif
      numpk = nwdvar
      nbit = native/numpk

#ifdef MPI
!     Only the master node performs the initial setup and
!     reading/writing
      if(.NOT.master) goto 120
#endif
!
!     --- get file names ---
!
      CALL mdfil
!     ----- READ THE NECESSARY DATA TO INITIATE THE RUN -----
#ifdef ROAR_CP
      CALL mdread(x,ix,ih,ifqnt,nquant,labels,mlabel)
#else
      CALL mdread(x,ix,ih)
#endif
#ifdef MPI
      if(master) then
#endif
! --- OPEN FILES FOR NHC RELATED OUTPUT
      IF(NHCPRNT.EQ.1) THEN
      CALL amopen(20,'nhcpress','N','F','W')
      CALL amopen(21,'nhcvolum','N','F','W')
      CALL amopen(22,'nhchprim','N','F','W')
      CALL amopen(23,'nhcaverag','N','F','W')
      CALL amopen(24,'nhcrattle','N','F','W')
      CALL amopen(25,'nhcblarea','N','F','W')
      END IF
#ifdef MPI
      endif
#endif
!     ----- EVALUATE SOME CONSTANTS -----
!
C     ONE = 1.0D0
      SMALL = 1.0D-4
      NRPT = NPM*NRP
      NR = NRPT+NSM*NRAM
      NR3 = 3*NR
      BELLY = IBELLY.GT.0
!
      IF (nbit .lt. 32 .and. nr .gt. 32767) THEN
        PRINT *, '  Too many atoms for 16 bit pairlist -'
        PRINT *, '    Recompile without ISTAR2'
        CALL mexit(1)
      ENDIF
!
!     --- this check important because of alloc of L45 & its use in runmd ---
!
      IF (NTP.GT.0.AND.IABS(NTB).NE.2) GOTO 1000
!
!     ----- READ COORDINATES AND VELOCITIES -----
!
      IF (NTC.NE.1) THEN
        NBONT = 0
        IF(NTC.EQ.3) NBONT = NBONT + NBONA
        NBONT = NBONT + NBONH
        IF(NBONT.GT.MXNHC2) THEN
         WRITE(6,55500) NBONT
55500    FORMAT(3X,'ARRAY BOUND OVERFLOW',/
     X          ,3X,'CHANGE PARAMETER MXNHC2 IN nhc.h TO >=',I8)
         call mexit(1)
        END IF
      END IF
      NLINK=0
      CALL GETCORNHC(NR,X(L30),X(L55),X(L40),X(L35),NTX,
     $               BOX,BETA,T,NTC,NTP,NLINK)
      CALL NHCBOX
!
#ifdef ROAR_CP
      if(ifqnt.eq.0) then
          if(ntc.eq.3) then
              do 100 i=1,(nbonh+mbona)
                  iqmshk(i) = 1
 100          continue
          else if(ntc.eq.2) then
              do 110 i=1,nbonh
                  iqmshk(i) = 1
 110          continue
              do 115 i=1,mbona
                  iqmshk(nbonh+i) = 0
 115          continue
          else if(ntc.eq.1) then
              do 122 i=1,nbonh+mbona
                  iqmshk(i) = 0
 122          continue
          else
              write(6,*)'NTC assigned impossible value'
              call mexit(1)
          end if
       end if

!     assign link atoms between quantum mechanical and molecular mechanical
!     atoms if quantum atoms are present
!
!     after assigning the link atoms delete all connectivity between the
!     QM atoms
!
      if(ifqnt.eq.1) then
!
!         write(6,*)'Some coords'
!         do 7 i=1,10
!             write(6,1999)(x(l30+(i-1)*3+j-1),j=1,3)
! 7       continue
 1999 format(5x,3(3x,f10.6))
          klink = nlink
          call link_atoms(mbona,x(L30),nlink,ix(i18),
     $          ix(i20),ix(i22),x(L30+3*natom),nquant,
     $          labels,natom,x(L20+natom),mmqmbo(1),mmqmbo(2),
     $          ix(i62+nr))
          if((imin.eq.0).and.(nlink.ne.0).and.(klink.eq.0)) then
              write(6,1997)
              write(6,1998)
              call mexit(1)
          end if
 1997 format(25X,'FATAL ERROR')
 1998 format(10X,'Link atoms MUST be optimized before MD run')
!         write(6,*)'More coords'
!         do 8 i=1,10
!             write(6,1999)(x(l30+(i-1)*3+j-1),j=1,3)
! 8       continue
!
          write(6,*)'nbonh,mbona,nbona ==> ',nbonh,mbona,nbona
          if(nbonh.gt.0) then
              call del_bond(nbonh,ix(i12),ix(i14),ix(i16),nquant,
     $                                                     labels)
          end if
!
          if(mbona.gt.0) then
              itemp = mbona
              call del_bond2(mbona,ix(i12+nbonh),ix(i14+nbonh),
     $                      ix(i16+nbonh),ix(i18),ix(i20),ix(i22),
     $                      nquant,labels)
              idiff = itemp - mbona
              nbona = nbona - idiff
!
!     adjust memory location pointers to reflect changes in bonding
!
              I18 = I12 + nbonh
              I20 = I14 + nbonh
              I22 = I16 + nbonh
          end if
          write(6,*)'nbonh,mbona,nbona ==> ',nbonh,mbona,nbona
!
!         now that all of the "qm bonds" have been deleted from the bond lists
!         need to reconstruct the SHAKE bond list
!
          dummy = 0.0d0
          call bshake(nbonh,nbona,0,ix(i16),x(l50),req,dummy)
!
          if(ntheth.gt.0) then
              call del_angl(ntheth,ix(i24),ix(i26),ix(i28),ix(i30),
     $                                              nquant,labels)
          end if
!
          if(ntheta.gt.0) then
              call del_angl(ntheta,ix(i32),ix(i34),ix(i36),ix(i38),
     $                                              nquant,labels)
```

```fortran
            end if
!
            write(6,*)'nphih ==> ',nphih
            if(nphih.gt.0) then
                call del_dihed(nphih,ix(i40),ix(i42),ix(i44),ix(i46),
     $                                        ix(i48),nquant,labels)
            end if
            write(6,*)'nphih ==> ',nphih
!
            write(6,*)'nphia,mphia ==> ',nphia,mphia
            if(nphia.gt.0) then
                call del_dihed(nphia,ix(i50),ix(i52),ix(i54),ix(i56),
     $                                        ix(i58),nquant,labels)
                mphia = nphia
            end if
            write(6,*)'nphia,mphia ==> ',nphia,mphia
!
!       set flags to run shake for bonds between mm atoms but not for
!       bonds between qm and mm atoms
!
!           write(6,*)'nbonh ==> ',nbonh
!           write(6,*)'mbona ==> ',mbona
            if(ntc.eq.3) then
                do 200 i=1,(nbonh+mbona)
                    iqmshk(i) = 1
 200            continue
            else if(ntc.eq.2) then
                do 210 i=1,nbonh
                    iqmshk(i) = 1
 210            continue
                do 215 i=1,mbona
                    iqmshk(nbonh+i) = 0
 215            continue
            else if(ntc.eq.1) then
                do 230 i=1,nbonh+mbona
                    iqmshk(i) = 0
 230            continue
            else
                write(6,*)'NTC assigned impossible value'
                call mexit(1)
            end if
            write(6,*)'mbona,nbonh ==> ',mbona,nbonh
            if((nbonh.gt.0).and.(ntc.gt.1))then
                call ifshk(nbonh,ix(i12),ix(i14),nquant,labels,iqmshk)
           end if
!
!           write(6,*)'mbona ==> ',mbona
            if((mbona.gt.0).and.(ntc.gt.2))then
                call ifshk(mbona,ix(i18),ix(i20),nquant,labels,
     $                                              iqmshk(nbonh+1))
            end if
!
!           Output any PMF information.
!
            if(ipert.ne.0) then
              if(npert.ne.0)then
                write(6,'(/" IN MINMD4, ATOMS INVOLVED IN THE PMF:")')
                do 10 i=1,npert
                  write(6,'(/" GROUP ",i2,":")') i
                  write(6,'(16i5)') (iatms(j),j=istrt(i),iend(i))
                  if(lnkend(i).gt.iend(i))then
                    write(6,'(" LINK ATOMS:")')
                    write(6,'(16i5)') (iatms(j),j=iend(i)+1,lnkend(i))
                  endif
 10             continue
```

```
!
!             Compute and output initial center of mass or bonded atom
!             coordinates.  If user has specified all zeros for sxcm,
!             sycm, and szcm, then stop execution.
!
              call cmwrit(x(L30),x(L20),istop)
              if (istop.ne.0) then
                 write(6,*)'Error in cmwrit called from sander.F'
                 call mexit(1)
              endif
            endif
          endif
!
!         Assign any QM constraints.
!
          if(nquant.gt.1)then
            if(.not.do_scf)then
!
!             A slow growth FEP calculation to zero out QM van der
!             Waals parameters has been requested, and there is no
!             electrostatic coupling of the MM and QM systems.  In
!             this case, no QM scf calculations will be done, so the
!             QM system (i.e., solute) needs to be locked in a rigid
!             conformation to keep it from falling apart.  Generate
!             the appropriate constraints.
!
              call rigid(x(L30),nquant,labels)
            else
!
!             See if there is a user-defined list of constraints
!             in the file constraint.dat.
!
              call rdcnst(x(L30),ierror)
              if (ierror.ne.0) then
                 write(6,*)'Error in rdcnst called from sander.F'
                 call mexit(1)
              endif

            endif
          endif
!
!         See if the user has constrained any atoms to lie in a plane.
!
          call rdpln(ierror)
          if (ierror.ne.0) then
             write(6,*)'Error in rdpln called from sander.F'
             call mexit(1)
          endif
!
!         See if the user has constrained sets of atoms to have the
!         same bond length.
!
          call rdsym(ierror)
          if (ierror.ne.0) then
             write(6,*)'Error in cmwrit called from sander.F'
             call mexit(1)
          endif
!
!
!     zero out the charges on the quantum mechanical atoms
!
          do 235 i=1,nquant
              index = L15 + labels(i) - 1
              x(index) = 0.0d0
 235      continue
```

```
      end if
!
#endif
!
!     --- Set up principal (marker) atom list for res based imaging ---
!
      IF (ntb .ne. 0) THEN
        CALL setmrk(natom,nres,ix(i02),x(L30),ix(i01))
      ENDIF
!
      IF(INIT.EQ.4.AND.NTX.LT.4) INIT = 3
!
! Set up the solute/solvent pointers:
!
            CALL SOLPNT(NSOLW      ,IBGWAT     ,IPTRES     ,IPTSOL     ,
     *       NATRCM     ,IPTATM     ,IFTRES     ,ISOLVP     ,NATOM      ,
     *       NRES       ,NSOLUT     ,NTT        ,NSPSOL     ,NSPSTR     ,
     *       IX(I02)     ,6)
!
!     ----- OPEN THE DATA DUMPING FILES AND POSITION IT DEPENDING
!           ON THE TYPE OF RUN -----
!
      CALL OUTOPN
!
! ----------------------------------------------------------------------
! Main MD loop over NRUN
! ----------------------------------------------------------------------
!
  120 CONTINUE
      CALL get_time(wtim0)


  220 CONTINUE
      SKIP = .FALSE.
      NRU = NRU+1
!
      CALL MDBOX
!
#ifdef MPI
!     ...send all data needed to other nodes, now that master has it
!
C     ONE = 1.0D0
      SMALL = 1.0D-4
      NRPT = NPM*NRP
      NR = NRPT+NSM*NRAM
      NR3 = 3*NR
      BELLY = IBELLY.GT.0
      CALL startup(x,ix,ih)
!
      IF (master) write(6, '(1x,a,i4,a,/)')
     .      'Running AMBER ROAR MPI version on ',numtasks, ' nodes'
#endif
      call gaussfg_init(natom,mbona,ix(i18),ix(i20),nquant,labels,nlink) !wpp
!
! ----------------------------------------------------------------------
! Now do the dynamics or minimization.
! ----------------------------------------------------------------------
!
! Dynamics:
!
      IF (IMIN.EQ.0) THEN
!
#ifdef ROAR_CP
      CALL RUNMD(x,ix,ih,X(L30),X(L20),X(L35),X(L40),X(L45),X(L55),
     +            X(L50),X(L95),IX(I70),X(L75),erstop,ifqnt,nquant,
     +      labels,mlabel,nlink,mmqmbo,iqmshk,iqmres)
#else

        CALL RUNMD(x,ix,ih,X(L30),X(L20),X(L35),X(L40),X(L45),X(L55),
     +          X(L50),X(L95),IX(I70),X(L75),erstop)
#endif

        INIT = 4
!
! Write the restart file:
!
#ifdef MPI
        IF (master) THEN
#endif
!
          CALL MDWRITNHC(NPM,NRP,NR,NRES,NTXO,NTR,NTP,X(L30),X(L40),
     +          X(L55),BOX,ih(m04),ih(m02),IX(I02),T,NTC,nlink)
!
#ifdef MPI
        ENDIF
#endif
!
! Check time remaining. Exit if max. time exceeded.
!
        CALL TIMIT(1,SKIP,6)
! JV Allow all processors access to this write stmt
! If CPU time is actually up this will indicate which processor
! timed out first giving possible clue to load balance problem
        IF (SKIP) WRITE(6,1020)
!
! If time not exceeded, and specified NSTLIM runs not completed, go do another:
!
        IF (.NOT. SKIP .AND. NRU .LT.NRUN) GO TO 220
!
      ELSE
!
! Minimization:
!
#ifdef ROAR_CP
         CALL RUNMIN(x,ix,ih,X(L30),X(L35),X(L40),ih(m04),ih(m02),
     +            IX(I02),IX(I12),IX(I14),X(L50),X(L20),IX(I62),
     +            X(L95),IX(I70),ERSTOP,CONVGD,ene,ifqnt,nquant,
     +      labels,mlabel,nlink,mmqmbo,iqmshk,iqmres)
#else
        CALL RUNMIN(x,ix,ih,X(L30),X(L35),X(L40),ih(m04),ih(m02),
     +           IX(I02),IX(I12),IX(I14),X(L50),X(L20),IX(I62),
     +           X(L95),IX(I70),ERSTOP,CONVGD,ene)
#endif

#ifdef MPI
        IF (master) THEN
#endif
!
! Write the restart file:
!
        CALL MINRIT(NRES,X(L30),ih(m04),ih(m02),IX(I02),nlink)
        CALL TIMIT(1,SKIP,6)
#ifdef MPI
        ENDIF
#endif

!
      ENDIF
!
!     -- calc time spent running vs setup
```

```
!
      CALL get_time(wtim1)
      timsts(NUMSTS) = wtim1 - wtim0
!
! When run is over, call profil to write timings:
!
#ifdef MPI
#ifdef PROFILE
      CALL profile_mpi
#endif
      if (master) then
#endif
      CALL CPU_PROFILE
#ifdef MPI
      endif
#endif
      CALL mexit(0)
!
 1000 WRITE(6,1010)
 1010 FORMAT(/ /,'INPUT NTP/NTB INCONSISTENT')
 1020 FORMAT(/ /5X,'CPU TIME LIMIT EXCEEDED')
 1030 FORMAT(/5X,'VELOCITIES HAVE BEEN RESCALED',/)
      CALL mexit(1)
      END
! ==========================================================
      subroutine del_bond(nbonds,ib,jb,icb,nquant,labels)
!
!     This subroutine deletes bonds between pairs of quantum
!     atoms.  The pointer to the appropriate bond constants
!     is also adjusted to reflect this change.
!
      implicit double precision (a-h,o-z)
!
      dimension ib(*),jb(*),icb(*),labels(*)
!
      k = 1
!     write(6,*)'In Del Bonds'
      do 100 i=1,nbonds
          i3 = ib(i)/3 + 1
          j3 = jb(i)/3 + 1
!         write(6,*)'ib,jb ==> ',(ib(i)/3+1),(jb(i)/3+1)
          ii = 0
          jj = 0
          do 110 j=1,nquant
              if(i3.eq.labels(j)) ii = 1
              if(j3.eq.labels(j)) jj = 1
 110      continue
          if(ii+jj.ne.2) then
              icb(k) = icb(i)
              ib(k) = ib(i)
              jb(k) = jb(i)
              k = k + 1
          end if
 100  continue
      nbonds = k - 1
!     write(6,*)'At end of del_bonds'
!     do 200 i=1,nbonds
!         i3 = ib(i)/3 + 1
!         j3 = jb(i)/3 + 1
!         write(6,*)'ib,jb ==> ',(ib(i)/3+1),(jb(i)/3+1)
!200  continue
      return
      end
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
```

```fortran
!-----------------------------------------------------------------------
      subroutine del_bond2(nbonds,newib,newjb,newicb,ib,jb,icb,
     $                                          nquant,labels)
!
!     This subroutine deletes bonds between pairs of quantum
!     atoms.  The pointer to the appropriate bond constants
!     is also adjusted to reflect this change.
!
      implicit double precision (a-h,o-z)
!
      dimension ib(*),jb(*),icb(*),labels(*)
      dimension newib(*),newjb(*),newicb(*)
!
      k = 1
!     write(6,*)'In Del Bonds'
      do 100 i=1,nbonds
          i3 = ib(i)/3 + 1
          j3 = jb(i)/3 + 1
!         write(6,*)'ib,jb ==> ',(ib(i)/3+1),(jb(i)/3+1)
          ii = 0
          jj = 0
          do 110 j=1,nquant
              if(i3.eq.labels(j)) ii = 1
              if(j3.eq.labels(j)) jj = 1
 110      continue
          if(ii+jj.ne.2) then
              newicb(k) = icb(i)
              newib(k) = ib(i)
              newjb(k) = jb(i)
              k = k + 1
          end if
 100  continue
      nbonds = k - 1
!     write(6,*)'At end of del_bonds'
!     do 200 i=1,nbonds
!         i3 = ib(i)/3 + 1
!         j3 = jb(i)/3 + 1
!         write(6,*)'ib,jb ==> ',(ib(i)/3+1),(jb(i)/3+1)
!200  continue
      return
      end
!#######################################################
      subroutine del_angl(nangl,ib,jb,kb,icb,nquant,labels)
!     This subroutine deletes angles involving trios of quantum
!     atoms.  The corresponding constants and equilibrium
!     values are also removed from the list.
!
      implicit double precision (a-h,o-z)
!
      dimension ib(*),jb(*),kb(*),icb(*),labels(*)
!
      m = 1
      do 100 i=1,nangl
          i3 = ib(i)/3 + 1
          j3 = jb(i)/3 + 1
          k3 = kb(i)/3 + 1
          ii = 0
          jj = 0
          kk = 0
          do 110 j=1,nquant
              if(i3.eq.labels(j)) ii = 1
              if(j3.eq.labels(j)) jj = 1
              if(k3.eq.labels(j)) kk = 1
 110      continue
          if(ii+jj+kk.ne.3) then
```

```fortran
                icb(m) = icb(i)
                ib(m) = ib(i)
                jb(m) = jb(i)
                kb(m) = kb(i)
                m = m + 1
            end if
 100  continue
      nangl = m - 1
      return
      end
!#################################################
      subroutine del_dihed(ndihed,ib,jb,kb,lb,icb,nquant,labels)
!
!     This subroutine deletes dihedral angles involving quartets
!     of quantum atoms.
!
      implicit double precision (a-h,o-z)
!
      dimension ib(*),jb(*),kb(*),lb(*),icb(*),labels(*)
!
      m = 1
      write(6,*)'ndihed ==>',ndihed
      do 100 i=1,ndihed
          i3 = ib(i)/3 + 1
          j3 = jb(i)/3 + 1
          k3 = iabs(kb(i))/3 + 1
          l3 = iabs(lb(i))/3 + 1
!         write(6,*)'i3,j3,k3,l3 ==> ',i3,j3,k3,l3
!         if(l3.lt.0) l3 = iabs(l3) + k3
!         write(6,*)'i3,j3,k3,l3 ==> ',i3,j3,k3,l3
          ii = 0
          jj = 0
          kk = 0
          ll = 0
          do 110 j=1,nquant
              if(i3.eq.labels(j)) ii = 1
              if(j3.eq.labels(j)) jj = 1
              if(k3.eq.labels(j)) kk = 1
              if(l3.eq.labels(j)) ll = 1
 110      continue
!         write(6,*)'ii,jj,kk,ll',ii,jj,kk,ll
          if(ii+jj+kk+ll.ne.4) then
              icb(m) = icb(i)
              ib(m) = ib(i)
              jb(m) = jb(i)
              kb(m) = kb(i)
              lb(m) = lb(i)
              m = m + 1
          end if
 100  continue
      ndihed = m - 1
      write(6,*)'ndihed ==>',ndihed
      return
      end

```