

```

1  ! --- MODIFIED TO INCLUDE NHC-EWALD-COUPLED POTENTIAL STUFF
2  ! --- NMR refinement are removed
3  !
4  !   PROGRAM SANDER
5  !
6  !       SANDER, version 4.1 (with NHC and EWALD)
7  !
8  ! The Molecular Dynamics/NMR Refinement/Modeling Module of the AMBER Package.
9  !
10 !   This is a modified version of the AMBER 3.0, Rev. A MD Module which
11 !   includes an extensive suite of utilites for use with NMR refinement,
12 !   other modeling calculations, as well as other new options within
13 !   the bulk of the MD program (minimization is also included).
14 !
15 !   The NMR refinement/modeling suite, hooks thereto, and the new options
16 !   were written by
17 !
18 !       David A. Pearlman (UCSF)
19 !       David A. Case (Scripps) &
20 !       Ping Yip      (Scripps)
21 !
22 !   Version 4.1 also include the capability to carry out polarization
23 !   calculations. The polarizability code was written by Jim Caldwell
24 !   and Liem X. Dang (UCSF). Truncated octahedral periodicity was
25 !   added by Thomas Huber of Ludwig Maximilian Universitaet Muenchen.
26 !
27 !   Additional changes for version 4.1 are listed below.
28 !
29 !   Revision A of version 3 of AMBER MD, on which this program was built,
30 !   was by George L. Seibel. Better vectorized nonbond routines
31 !   in version 3A were written by Rad Olson and Bill Swope (IBM).
32 !
33 !   Version 3 of AMBER MD was written by U.C. Singh and Peter A. Kollman,
34 !   adapting significantly from GROMOS83 by Wilfred van Gunsteren.
35 !
36 !*****
37 !                               AMBER                               **
38 !                               **                                  **
39 ! Copyright (c) 1986, 1991 Regents of the University of California **
40 !                               All Rights Reserved.              **
41 !                               **                                  **
42 ! This software provided pursuant to a license agreement containing **
43 ! restrictions on its disclosure, duplication, and use. This software **
44 ! contains confidential and proprietary information, and may not be **
45 ! extracted or distributed, in whole or in part, for any purpose **
46 ! whatsoever, without the express written permission of the authors. **
47 ! This notice, and the associated author list, must be attached to **
48 ! all copies, or extracts, of this software. Any additional **
49 ! restrictions set forth in the license agreement also apply to this **
50 ! software. **
51 !*****
52 !
53 ! Significant changes for Version 4.1:
54 !
55 !     1) Inclusion of polarization code.
56 !        (J. Caldwell and L. Dang)
57 !     2) Use of fast analytical shake for 3 point waters.
58 !        (D. Pearlman and S. Miyamoto)
59 !     3) New much faster routines to handle TIP3P-TIP3P water interactions
60 !        (D. Case and D. Pearlman)
61 !     4) Inclusion of standard and time-averaged J-coupling restraints
62 !        (D. Pearlman)
63 !     5) New methods for NMR Intensity refinement and ring current calcs.
64 !        (D. Case)
65 !     6) Allow for dual non-bonded cutoffs.

```

```

66 !      (D. Pearlman)
67 !      7) New VLIMIT option to limit max. atomic velocity.
68 !      (D. Pearlman)
69 !      8) Incorporation of PEACS constant nrg contour conf. search capability
70 !      (D. Case)
71 !      9) Plus various minor modifications for clearer output, cleanup,
72 !         to incorporate bugfixes, etc.
73 !      10) New code for truncated octahedral periodic boundary conditions
74 !          (Thomas Huber, Ludwig Maximilian Universitaet Muenchen,
75 !           email: thuber@Physik.TU-Muenchen.de) and reorganization/relocation
76 !           of imaging routines to period.f (B. Ross).
77 !
78 ! Changes for Version 4 (NMR):
79 !
80 !     Aside from the various hooks required to
81 !     integrate the NMR package,
82 !     1) the nonbonded code in nonbon and ephi
83 !        was modified to allow a "soft repulsion" non-bonded potential in
84 !        place of 6-12 or 10-12 vdw interactions;
85 !     2) several new temperature-coupling options were added in RUNMD.
86 !        These will be particularly useful when carrying out simulations
87 !        where the internal energy of the molecular system is changing
88 !        very quickly (such as in some MD/NMR refinement schemes).
89 !     3) The "NMR" package itself, which allows
90 !        a large number of simulation protocols appropriate for NMR/MD
91 !        refinements and general modeling work, and offers a relatively
92 !        flexible and easy-to-use interface. See the SANDER refinement
93 !        manual for more details.
94 !
95 ! =====
96 !
97 !     implicit double precision (a-h,o-z)
98 !
99 !     LOGICAL SKIP, BELLY, erstop
100 #include "files.h"
101 #include "sizes.h"
102 #include "memory.h"
103 #include "box.h"
104 #include "md.h"
105 #include "parms.h"
106 #ifdef ROAR_CP
107 #include "cp.h"
108 #endif
109
110 #ifdef MPI
111 #include "mpif.h"
112 #include "parallel.h"
113 #endif
114
115 #include "nhc.h"
116 !
117 #include "ewald.h"
118 #include "pmedim.h"
119 #include "pme.h"
120 !
121     dimension X(MAXREA)
122     integer ix(MAXINT)
123     integer ih(MAXHOL)
124     dimension ene(30)
125     real wtim0, wtim1
126
127 !
128 ! Initialize the cpu timer. Needed for machines where returned cpu times
129 ! are relative.
130 !

```

```

131     CALL TIMIT(3,SKIP,6)
132
133 #ifdef HP
134 !
135 !   --- set up certain underflow operations
136 !
137     on double precision underflow call trapud
138     on real underflow call trapu
139 #endif
140 #ifdef AIX
141     call setrteopts('namelist=old')
142 #endif
143 !
144 !   --- Code Configuration Section ---
145 !
146 !   Revision A code is designed to port easily to machines of
147 !   varying wordsize. We have changed the memory mapping scheme
148 !   of version 3.0 to improve portabilty. Instead of one large
149 !   array there are now three separate arrays. The following points
150 !   are notable:
151 !   1) Three arrays X, IH, and IX are passed as arguments instead
152 !       of COMMON. These arrays are for Reals, Hollerith ints, and
153 !       Integers, respectively. The Reals may be 32 or 64 bit, and
154 !       Integers may be 32 or 64 bit as well. The Hollerith int
155 !       and numerical int arrays share the same starting address,
156 !       by virtue of the equivalence in main. This is *not* non-portable,
157 !       don't get excited. All of the structural arrays in the program
158 !       are mapped into the three large arrays, and are referenced
159 !       by the offsets passed in commons MEMLA for reals, MEMLB and
160 !       MEMLD for ints, and MEMLC for Hollerith data.
161 !   2) Two variables are used to describe the packing of the
162 !       nonbonded pairlist. IPACK is set to 1 if explicit word-
163 !       packing routines are called, as on Cray or FPS. Otherwise
164 !       IPACK = 0. NWDVAR is the number of NB pair pointers that is
165 !       held in a default integer word. This will usually be 4 for
166 !       the Cray, 2 or 4 for the FPS, and on a 32 bit machine will
167 !       be 1 if the entire word is used, or 2 if an integer*2 declaration
168 !       is used for the pairlist.
169 !       NATIVE is the number of bits in a default integer on the target
170 !       machine. It is usually 32 or 64.
171 !   3) Memory Requirements: The three parameters MAXREA, MAXINT,
172 !       and MAXDUP are used to control memory use.
173 !
174 !       Array      Use      Parameter      Typical Value
175 !       X          floating pt  MAXREA        ~ 23 * Natom
176 !       IX         Integers    MAXINT        ~ 150 * Natom
177 !       (various) dihedral dup  MAXDUP        0 - 1000, data dependent
178 !
179 !       The typical values given are only rough estimates. The Integer
180 !       memory requirement consists of a "static" requirement, which
181 !       is topology dependent and does not vary throughout the run,
182 !       and a variable amount for the nonbonded pairlist pointers.
183 !       The value of ~150*Natom includes both the static requirement
184 !       and the pairlist requirement for a "typical" system assuming
185 !       a full word is used to store a pairlist pointer. To determine
186 !       the actual Integer memory requirement, add the static requirement
187 !       reported at the start of a run to the pairlist requirement. The
188 !       pairlist requirement is the total number of nonbonded pairs (this
189 !       is geometry and cutoff dependent) divided by NWDVAR. If IPACK
190 !       is not 0, you should add Natom. Since the pairlist can grow
191 !       during a run (and often does) it is a good idea to increase the
192 !       room for it by ~10%. The maximum number of nonbonded pairs
193 !       for the value of MAXINT used will be reported at the top of the
194 !       output.
195 !

```

```
196 !           IPACK=0:  MAXINT = Static Int + (NPAIRS/NWDVAR)*1.1
197 !           IPACK=1:  MAXINT = Static Int + (NPAIRS/NWDVAR)*1.1+NATOM
198 !
199 !           All cases: MAXREA = reported static output.
200 !
201 !           Dihedrals that have more than one fourier term will have their
202 !           pointers duplicated for the vectorized dihedral routine. This
203 !           is done twice; once for heavy atom dihedrals and once for dihedrs
204 !           involving H-atoms.  MAXDUP must be at least as large as the
205 !           larger number reported in the output.  The actual amount of space
206 !           allocated is 10*MAXDUP, so it should not be set too large in a
207 !           tight memory environment.
208 !
209 #ifdef MPI
210 !           Set up parallel execution
211 !
212 !           CALL mpi_init(ierr)
213 !           CALL mpi_comm_rank(MPI_COMM_WORLD,mytaskid,ierr)
214 !           CALL mpi_comm_size(MPI_COMM_WORLD,numtasks,ierr)
215 !
216 !           Make PE 0 the master
217 !           master = mytaskid.EQ.0
218 #endif
219 !           NRU = 0
220 !           NHCSETQ = .TRUE.
221 !
222 !           erstop = .false.
223 !           --- generic packing scheme ---
224 !           nwdvar = 1
225 !           native = 32
226 !           nlink =0
227 !
228 #ifdef ISTAR2
229 !           --- Int*2 packing scheme ---
230 !           nwdvar = 2
231 #endif
232 !           numpk = nwdvar
233 !           nbit = native/numpk
234
235 #ifdef MPI
236 !           Only the master node performs the initial setup and
237 !           reading/writing
238 !           if(.NOT.master) goto 120
239 #endif
240 !
241 !           --- get file names ---
242 !
243 !           CALL mdfil
244 !           ----- READ THE NECESSARY DATA TO INITIATE THE RUN -----
245 #ifdef ROAR_CP
246 !           CALL mdread(x,ix,ih,ifqnt,nquant,labels,mlabel)
247 #else
248 !           CALL mdread(x,ix,ih)
249 #endif
250 #ifdef MPI
251 !           if(master) then
252 #endif
253 !           --- OPEN FILES FOR NHC RELATED OUTPUT
254 !           IF(NHCPRNT.EQ.1) THEN
255 !           CALL amopen(20,'nhcpress','N','F','W')
256 !           CALL amopen(21,'nhcvolum','N','F','W')
257 !           CALL amopen(22,'nhchprim','N','F','W')
258 !           CALL amopen(23,'nhcaverag','N','F','W')
259 !           CALL amopen(24,'nhcrattle','N','F','W')
260 !           CALL amopen(25,'nhcblarea','N','F','W')
```

```

261     END IF
262 #ifdef MPI
263     endif
264 #endif
265 !     ----- EVALUATE SOME CONSTANTS -----
266 !
267 C     ONE = 1.0D0
268     SMALL = 1.0D-4
269     NRPT = NPM*NRP
270     NR = NRPT+NSM*NRAM
271     NR3 = 3*NR
272     BELLY = IBELLY.GT.0
273 !
274     IF (nbit .lt. 32 .and. nr .gt. 32767) THEN
275         PRINT *, ' Too many atoms for 16 bit pairlist -'
276         PRINT *, '   Recompile without ISTAR2'
277         CALL mexit(1)
278     ENDIF
279 !
280 !     --- this check important because of alloc of L45 & its use in runmd ---
281 !
282     IF (NTP.GT.0.AND.IABS(NTB).NE.2) GOTO 1000
283 !
284 !     ----- READ COORDINATES AND VELOCITIES -----
285 !
286     IF (NTC.NE.1) THEN
287         NBONT = 0
288         IF(NTC.EQ.3) NBONT = NBONT + NBOA
289         NBONT = NBONT + NBONH
290         IF(NBONT.GT.MXNHC2) THEN
291             WRITE(6,55500) NBONT
292 55500     FORMAT(3X,'ARRAY BOUND OVERFLOW',/
293 X           ,3X,'CHANGE PARAMETER MXNHC2 IN nhc.h TO >=',I8)
294             call mexit(1)
295         END IF
296     END IF
297     NLINK=0
298     CALL GETCORNHC(NR,X(L30),X(L55),X(L40),X(L35),NTX,
299 $               BOX,BETA,T,NTC,NTP,NLINK)
300     CALL NHCBOX
301 !
302 #ifdef ROAR_CP
303     if(ifqnt.eq.0) then
304         if(ntc.eq.3) then
305             do 100 i=1,(nbonh+mbona)
306                 iqmshk(i) = 1
307 100         continue
308         else if(ntc.eq.2) then
309             do 110 i=1,nbonh
310                 iqmshk(i) = 1
311 110         continue
312             do 115 i=1,mbona
313                 iqmshk(nbonh+i) = 0
314 115         continue
315         else if(ntc.eq.1) then
316             do 122 i=1,nbonh+mbona
317                 iqmshk(i) = 0
318 122         continue
319         else
320             write(6,*)'NTC assigned impossible value'
321             call mexit(1)
322         end if
323     end if
324 !
325 !     assign link atoms between quantum mechanical and molecular mechanical

```

```
326 ! atoms if quantum atoms are present
327 !
328 ! after assigning the link atoms delete all connectivity between the
329 ! QM atoms
330 !
331 ! if(ifqnt.eq.1) then
332 !
333 ! write(6,*)'Some coords'
334 ! do 7 i=1,10
335 ! write(6,1999)(x(l30+(i-1)*3+j-1),j=1,3)
336 ! 7 continue
337 1999 format(5x,3(3x,f10.6))
338 klink = nlink
339 call link_atoms(mbona,x(l30),nlink,ix(i18),
340 $ ix(i20),ix(i22),x(l30+3*natom),nquant,
341 $ labels,natom,x(l20+natom),mmqambo(1),mmqambo(2),
342 $ ix(i62+nr))
343 if((imin.eq.0).and.(nlink.ne.0).and.(klink.eq.0)) then
344 write(6,1997)
345 write(6,1998)
346 call mexit(1)
347 end if
348 1997 format(25X,'FATAL ERROR')
349 1998 format(10X,'Link atoms MUST be optimized before MD run')
350 ! write(6,*)'More coords'
351 ! do 8 i=1,10
352 ! write(6,1999)(x(l30+(i-1)*3+j-1),j=1,3)
353 ! 8 continue
354 !
355 write(6,*)'nbonh,mbona,nbona ==> ',nbonh,mbona,nbona
356 if(nbonh.gt.0) then
357 call del_bond(nbonh,ix(i12),ix(i14),ix(i16),nquant,
358 $ labels)
359 end if
360 !
361 if(mbona.gt.0) then
362 itemp = mbona
363 call del_bond2(mbona,ix(i12+nbonh),ix(i14+nbonh),
364 $ ix(i16+nbonh),ix(i18),ix(i20),ix(i22),
365 $ nquant,labels)
366 idiff = itemp - mbona
367 nbona = nbona - idiff
368 !
369 ! adjust memory location pointers to reflect changes in bonding
370 !
371 I18 = I12 + nbonh
372 I20 = I14 + nbonh
373 I22 = I16 + nbonh
374 end if
375 write(6,*)'nbonh,mbona,nbona ==> ',nbonh,mbona,nbona
376 !
377 ! now that all of the "qm bonds" have been deleted from the bond lists
378 ! need to reconstruct the SHAKE bond list
379 !
380 dummy = 0.0d0
381 call bshake(nbonh,nbona,0,ix(i16),x(l50),req,dummy)
382 !
383 if(ntheth.gt.0) then
384 call del_angl(ntheth,ix(i24),ix(i26),ix(i28),ix(i30),
385 $ nquant,labels)
386 end if
387 !
388 if(ntheta.gt.0) then
389 call del_angl(ntheta,ix(i32),ix(i34),ix(i36),ix(i38),
390 $ nquant,labels)
```

```

391         end if
392     !
393         write(6,*)'nphih ==> ',nphih
394         if(nphih.gt.0) then
395             call del_dihed(nphih,ix(i40),ix(i42),ix(i44),ix(i46),
396                 $           ix(i48),nquant,labels)
397         end if
398         write(6,*)'nphih ==> ',nphih
399     !
400         write(6,*)'nphia,mphia ==> ',nphia,mphia
401         if(nphia.gt.0) then
402             call del_dihed(nphia,ix(i50),ix(i52),ix(i54),ix(i56),
403                 $           ix(i58),nquant,labels)
404             mphia = nphia
405         end if
406         write(6,*)'nphia,mphia ==> ',nphia,mphia
407     !
408     ! set flags to run shake for bonds between mm atoms but not for
409     ! bonds between qm and mm atoms
410     !
411     ! write(6,*)'nbonh ==> ',nbonh
412     ! write(6,*)'mbona ==> ',mbona
413     ! if(ntc.eq.3) then
414         do 200 i=1,(nbonh+mbona)
415             iqmshk(i) = 1
416     200     continue
417         else if(ntc.eq.2) then
418             do 210 i=1,nbonh
419                 iqmshk(i) = 1
420     210     continue
421             do 215 i=1,mbona
422                 iqmshk(nbonh+i) = 0
423     215     continue
424         else if(ntc.eq.1) then
425             do 230 i=1,nbonh+mbona
426                 iqmshk(i) = 0
427     230     continue
428         else
429             write(6,*)'NTC assigned impossible value'
430             call mexit(1)
431         end if
432         write(6,*)'mbona,nbonh ==> ',mbona,nbonh
433         if((nbonh.gt.0).and.(ntc.gt.1))then
434             call ifshk(nbonh,ix(i12),ix(i14),nquant,labels,iqmshk)
435         end if
436     !
437     ! write(6,*)'mbona ==> ',mbona
438     ! if((mbona.gt.0).and.(ntc.gt.2))then
439         call ifshk(mbona,ix(i18),ix(i20),nquant,labels,
440             $           iqmshk(nbonh+1))
441     !
442     !
443     ! Output any PMF information.
444     !
445     if(ipert.ne.0) then
446         if(npert.ne.0)then
447             write(6,'(/" IN MINMD4, ATOMS INVOLVED IN THE PMF:")')
448             do 10 i=1,npert
449                 write(6,'(/" GROUP ",i2,":")') i
450                 write(6,'(16i5)') (iatms(j),j=istrt(i),iend(i))
451                 if(lnkend(i).gt.iend(i))then
452                     write(6,'(" LINK ATOMS:")')
453                     write(6,'(16i5)') (iatms(j),j=iend(i)+1,lnkend(i))
454                 endif
455     10     continue

```

```
456 !
457 !       Compute and output initial center of mass or bonded atom
458 !       coordinates.  If user has specified all zeros for sxcm,
459 !       sycm, and szcm, then stop execution.
460 !
461       call cmwrit(x(L30),x(L20),istop)
462       if (istop.ne.0) then
463         write(6,*)'Error in cmwrit called from sander.F'
464         call mexit(1)
465       endif
466     endif
467   endif
468 !
469 !       Assign any QM constraints.
470 !
471   if(nquant.gt.1)then
472     if(.not.do_scf)then
473       !
474       !       A slow growth FEP calculation to zero out QM van der
475       !       Waals parameters has been requested, and there is no
476       !       electrostatic coupling of the MM and QM systems.  In
477       !       this case, no QM scf calculations will be done, so the
478       !       QM system (i.e., solute) needs to be locked in a rigid
479       !       conformation to keep it from falling apart.  Generate
480       !       the appropriate constraints.
481       !
482       call rigid(x(L30),nquant,labels)
483     else
484       !
485       !       See if there is a user-defined list of constraints
486       !       in the file constraint.dat.
487       !
488       call rdcnst(x(L30),ierror)
489       if (ierror.ne.0) then
490         write(6,*)'Error in rdcnst called from sander.F'
491         call mexit(1)
492       endif
493     endif
494   endif
495 !
496 !       See if the user has constrained any atoms to lie in a plane.
497 !
498 !
499   call rdpln(ierror)
500   if (ierror.ne.0) then
501     write(6,*)'Error in rdpln called from sander.F'
502     call mexit(1)
503   endif
504 !
505 !       See if the user has constrained sets of atoms to have the
506 !       same bond length.
507 !
508   call rdsym(ierror)
509   if (ierror.ne.0) then
510     write(6,*)'Error in cmwrit called from sander.F'
511     call mexit(1)
512   endif
513 !
514 !
515 !       zero out the charges on the quantum mechanical atoms
516 !
517   do 235 i=1,nquant
518     index = L15 + labels(i) - 1
519     x(index) = 0.0d0
520 235 continue
```



```

521     end if
522     !
523 #endif
524     !
525     !   --- Set up principal (marker) atom list for res based imaging ---
526     !
527     IF (ntb .ne. 0) THEN
528         CALL setmrk(natom,nres,ix(i02),x(L30),ix(i01))
529     ENDIF
530     !
531     IF(INIT.EQ.4.AND.NTX.LT.4) INIT = 3
532     !
533     ! Set up the solute/solvent pointers:
534     !
535     CALL SOLPNT(NSOLW      ,IBGWAT      ,IPTRES      ,IPTSOL      ,
536 *   NATRCM      ,IPTATM      ,IFTRES      ,ISOLVP      ,NATOM      ,
537 *   NRES        ,NSOLUT      ,NTT        ,NSPSOL      ,NSPSTR      ,
538 *   IX(I02)     ,6)
539     !
540     ! ----- OPEN THE DATA DUMPING FILES AND POSITION IT DEPENDING
541     !           ON THE TYPE OF RUN -----
542     !
543     CALL OUTOPN
544     !
545     !-----
546     ! Main MD loop over NRUN
547     !-----
548     !
549     120 CONTINUE
550     CALL get_time(wtim0)
551     !
552     !
553     220 CONTINUE
554     SKIP = .FALSE.
555     NRU = NRU+1
556     !
557     CALL MDBOX
558     !
559 #ifdef MPI
560     !   ...send all data needed to other nodes, now that master has it
561     !
562     C   ONE = 1.0D0
563         SMALL = 1.0D-4
564         NRPT = NPM*NRP
565         NR = NRPT+NSM*NRAM
566         NR3 = 3*NR
567         BELLY = IBELLY.GT.0
568         CALL startup(x,ix,ih)
569     !
570     IF (master) write(6, '(1x,a,i4,a,/)' )
571     .   'Running AMBER ROAR MPI version on ',numtasks, ' nodes'
572 #endif
573     call gaussfg_init(natom,mbona,ix(i18),ix(i20),nquant,labels,nlink) !wpp
574     !
575     !-----
576     ! Now do the dynamics or minimization.
577     !-----
578     !
579     ! Dynamics:
580     !
581     IF (IMIN.EQ.0) THEN
582     !
583 #ifdef ROAR_CP
584     CALL RUNMD(x,ix,ih,X(L30),X(L20),X(L35),X(L40),X(L45),X(L55),
585 +           X(L50),X(L95),IX(I70),X(L75),erstop,ifqnt,nquant,

```

```
586     +      labels,mlabel,nlink,mmqambo,iqmshk,iqmres)
587 #else
588
589     CALL RUNMD(x,ix,ih,X(L30),X(L20),X(L35),X(L40),X(L45),X(L55),
590     +      X(L50),X(L95),IX(I70),X(L75),erstop)
591 #endif
592
593     INIT = 4
594 !
595 ! Write the restart file:
596 !
597 #ifdef MPI
598     IF (master) THEN
599 #endif
600 !
601     CALL MDWRITNHC(NPM,NRP,NR,NRES,NTXO,NTR,NTP,X(L30),X(L40),
602     +      X(L55),BOX,ih(m04),ih(m02),IX(I02),T,NTC,nlink)
603 !
604 #ifdef MPI
605     ENDIF
606 #endif
607 !
608 ! Check time remaining. Exit if max. time exceeded.
609 !
610     CALL TIMIT(1,SKIP,6)
611 ! JV Allow all processors access to this write stmt
612 ! If CPU time is actually up this will indicate which processor
613 ! timed out first giving possible clue to load balance problem
614     IF (SKIP) WRITE(6,1020)
615 !
616 ! If time not exceeded, and specified NSTLIM runs not completed, go do another:
617 !
618     IF (.NOT. SKIP .AND. NRU .LT. NRUN) GO TO 220
619 !
620     ELSE
621 !
622 ! Minimization:
623 !
624 #ifdef ROAR_CP
625     CALL RUNMIN(x,ix,ih,X(L30),X(L35),X(L40),ih(m04),ih(m02),
626     +      IX(I02),IX(I12),IX(I14),X(L50),X(L20),IX(I62),
627     +      X(L95),IX(I70),ERSTOP,CONVGD,ene,ifqnt,nquant,
628     +      labels,mlabel,nlink,mmqambo,iqmshk,iqmres)
629 #else
630     CALL RUNMIN(x,ix,ih,X(L30),X(L35),X(L40),ih(m04),ih(m02),
631     +      IX(I02),IX(I12),IX(I14),X(L50),X(L20),IX(I62),
632     +      X(L95),IX(I70),ERSTOP,CONVGD,ene)
633 #endif
634
635 #ifdef MPI
636     IF (master) THEN
637 #endif
638 !
639 ! Write the restart file:
640 !
641     CALL MINRIT(NRES,X(L30),ih(m04),ih(m02),IX(I02),nlink)
642     CALL TIMIT(1,SKIP,6)
643 #ifdef MPI
644     ENDIF
645 #endif
646
647 !
648     ENDIF
649 !
650 ! -- calc time spent running vs setup
```

```

651 !
652     CALL get_time(wtim1)
653     timsts(NUMSTS) = wtim1 - wtim0
654 !
655 ! When run is over, call profil to write timings:
656 !
657 #ifdef MPI
658 #ifdef PROFILE
659     CALL profile_mpi
660 #endif
661     if (master) then
662 #endif
663     CALL CPU_PROFILE
664 #ifdef MPI
665     endif
666 #endif
667     CALL mexit(0)
668 !
669 1000 WRITE(6,1010)
670 1010 FORMAT(/ /, 'INPUT NTP/NTB INCONSISTENT')
671 1020 FORMAT(/ /5X, 'CPU TIME LIMIT EXCEEDED')
672 1030 FORMAT(/5X, 'VELOCITIES HAVE BEEN RESCALED', /)
673     CALL mexit(1)
674     END
675 ! =====
676     subroutine del_bond(nbonds,ib,jb,icb,nquant,labels)
677 !
678 ! This subroutine deletes bonds between pairs of quantum
679 ! atoms. The pointer to the appropriate bond constants
680 ! is also adjusted to reflect this change.
681 !
682     implicit double precision (a-h,o-z)
683 !
684     dimension ib(*),jb(*),icb(*),labels(*)
685 !
686     k = 1
687 ! write(6,*)'In Del Bonds'
688     do 100 i=1,nbonds
689         i3 = ib(i)/3 + 1
690         j3 = jb(i)/3 + 1
691 ! write(6,*)'ib,jb ==> ',(ib(i)/3+1),(jb(i)/3+1)
692         ii = 0
693         jj = 0
694         do 110 j=1,nquant
695             if(i3.eq.labels(j)) ii = 1
696             if(j3.eq.labels(j)) jj = 1
697 110     continue
698         if(ii+jj.ne.2) then
699             icb(k) = icb(i)
700             ib(k) = ib(i)
701             jb(k) = jb(i)
702             k = k + 1
703         end if
704 100     continue
705     nbonds = k - 1
706 ! write(6,*)'At end of del_bonds'
707 ! do 200 i=1,nbonds
708 !     i3 = ib(i)/3 + 1
709 !     j3 = jb(i)/3 + 1
710 !     write(6,*)'ib,jb ==> ',(ib(i)/3+1),(jb(i)/3+1)
711 !200     continue
712     return
713     end
714 !-----
715 !-----

```

```

716 |-----
717 |     subroutine del_bond2(nbonds,newib,newjb,newicb,ib,jb,icb,
718 |     $                               nquant,labels)
719 | !
720 | !     This subroutine deletes bonds between pairs of quantum
721 | !     atoms. The pointer to the appropriate bond constants
722 | !     is also adjusted to reflect this change.
723 | !
724 |     implicit double precision (a-h,o-z)
725 | !
726 |     dimension ib(*),jb(*),icb(*),labels(*)
727 |     dimension newib(*),newjb(*),newicb(*)
728 | !
729 |     k = 1
730 | !     write(6,*)'In Del Bonds'
731 |     do 100 i=1,nbonds
732 |         i3 = ib(i)/3 + 1
733 |         j3 = jb(i)/3 + 1
734 | !         write(6,*)'ib,jb ==> ',(ib(i)/3+1),(jb(i)/3+1)
735 |         ii = 0
736 |         jj = 0
737 |         do 110 j=1,nquant
738 |             if(i3.eq.labels(j)) ii = 1
739 |             if(j3.eq.labels(j)) jj = 1
740 | 110         continue
741 |         if(ii+jj.ne.2) then
742 |             newicb(k) = icb(i)
743 |             newib(k) = ib(i)
744 |             newjb(k) = jb(i)
745 |             k = k + 1
746 |         end if
747 | 100     continue
748 |     nbonds = k - 1
749 | !     write(6,*)'At end of del_bonds'
750 | !     do 200 i=1,nbonds
751 | !         i3 = ib(i)/3 + 1
752 | !         j3 = jb(i)/3 + 1
753 | !         write(6,*)'ib,jb ==> ',(ib(i)/3+1),(jb(i)/3+1)
754 | !200     continue
755 |     return
756 |     end
757 | #####
758 |     subroutine del_angl(nangl,ib,jb,kb,icb,nquant,labels)
759 | !     This subroutine deletes angles involving trios of quantum
760 | !     atoms. The corresponding constants and equilibrium
761 | !     values are also removed from the list.
762 | !
763 |     implicit double precision (a-h,o-z)
764 | !
765 |     dimension ib(*),jb(*),kb(*),icb(*),labels(*)
766 | !
767 |     m = 1
768 |     do 100 i=1,nangl
769 |         i3 = ib(i)/3 + 1
770 |         j3 = jb(i)/3 + 1
771 |         k3 = kb(i)/3 + 1
772 |         ii = 0
773 |         jj = 0
774 |         kk = 0
775 |         do 110 j=1,nquant
776 |             if(i3.eq.labels(j)) ii = 1
777 |             if(j3.eq.labels(j)) jj = 1
778 |             if(k3.eq.labels(j)) kk = 1
779 | 110         continue
780 |         if(ii+jj+kk.ne.3) then

```

```
781         icb(m) = icb(i)
782         ib(m) = ib(i)
783         jb(m) = jb(i)
784         kb(m) = kb(i)
785         m = m + 1
786     end if
787 100 continue
788     nangl = m - 1
789     return
790 end
791 #####
792 subroutine del_dihed(ndihed,ib,jb,kb,lb,icb,nquant,labels)
793 !
794 !   This subroutine deletes dihedral angles involving quartets
795 !   of quantum atoms.
796 !
797 implicit double precision (a-h,o-z)
798 !
799 dimension ib(*),jb(*),kb(*),lb(*),icb(*),labels(*)
800 !
801 m = 1
802 write(6,*)'ndihed ==>',ndihed
803 do 100 i=1,ndihed
804     i3 = ib(i)/3 + 1
805     j3 = jb(i)/3 + 1
806     k3 = iabs(kb(i))/3 + 1
807     l3 = iabs(lb(i))/3 + 1
808 !   write(6,*)'i3,j3,k3,l3 ==> ',i3,j3,k3,l3
809 !   if(l3.lt.0) l3 = iabs(l3) + k3
810 !   write(6,*)'i3,j3,k3,l3 ==> ',i3,j3,k3,l3
811     ii = 0
812     jj = 0
813     kk = 0
814     ll = 0
815     do 110 j=1,nquant
816         if(i3.eq.labels(j)) ii = 1
817         if(j3.eq.labels(j)) jj = 1
818         if(k3.eq.labels(j)) kk = 1
819         if(l3.eq.labels(j)) ll = 1
820 110 continue
821 !   write(6,*)'ii,jj,kk,ll',ii,jj,kk,ll
822     if(ii+jj+kk+ll.ne.4) then
823         icb(m) = icb(i)
824         ib(m) = ib(i)
825         jb(m) = jb(i)
826         kb(m) = kb(i)
827         lb(m) = lb(i)
828         m = m + 1
829     end if
830 100 continue
831     ndihed = m - 1
832     write(6,*)'ndihed ==>',ndihed
833     return
834 end
835
```