```
!*************************************************************************
!                            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.                                                           **
!*************************************************************************
!
      SUBROUTINE RUNMIN(xx,ix,ih,X,FG,W,IGRAPH,LBRES,IPRES,IB,JB,CONP,
#ifdef ROAR_CP
     +             WINV,IGRP,SKIPS,NSP,ERSTOP,CONVGD,ENE,ifqnt,nquant,
     +       labels,mlabel,nlink,mmqmbo,iqmshk,iqmres)
#else
     +                  WINV,IGRP,SKIPS,NSP,ERSTOP,CONVGD,ENE)
#endif
!
!
!    This is the main minimization routine.  It primarily uses conjugate
!        gradients, but has an option for steepest descent minimization
!        as well.
!
!
      implicit double precision (a-h,o-z)

!
#ifdef ROAR_CP
# include "cp.h"
#endif
      LOGICAL ERSTOP,CONVGD,SKIP,NEWSTR,STEEP,BELLY,SKIPS
      double precision BETA, CRITS, DDSPLN, DFPR, DXST, DXSTH, DXSTM
      double precision F, FCH, FINIT, FMIN, FNQ, FOLD, GAMA, GAMDEN
      double precision GINIT, GMIN, GNEW, GSPLN, GSQRD, SBOUND, STEP
      double precision STEPCH, STMIN, SUM, WORK, DFPRED
!
#ifdef MPI
!     ========================== AMBER/MPI ===========================
!
!     NOTE: this routine contains MPI wrappers to all the I/O
!     routines to allow *only* the master to execute the code.
!     In addition, on entry fmin is set to zero in the T3D version
!     to prevent a core dump.
!
# include "parallel.h"
!     ======================== END AMBER/MPI =========================
#endif
!
!
!     ============================ EWALD ============================
!     ...for the EWALD version the only real changes in this routine
!     are that subroutine wrap_molecules() is called instead of SHIAG()
!     ========================== END EWALD ==========================
!
#include "md.h"
#include "box.h"
#include "files.h"
#include "memory.h"
!
      DIMENSION X(*),FG(*),W(*),IGRAPH(*),LBRES(*),IPRES(*)
      DIMENSION IB(*),JB(*),IGRP(*),CONP(*),WINV(*),SKIPS(*)
      DIMENSION NSP(*)
      DIMENSION ENE(30),VIR(4)
      dimension xx(*), ix(*), ih(*)
!
!  for lbfgs
```

```
      logical diagco
      dimension iprint(2)
!
      DATA MAXLIN,MXFCON,KSTCYC/ 10,  4 ,  4 /
      DATA DXSTM,CRITS,DFPRED/1.0D-05,1.0D-06,1.D0/
!
!     ----- EVALUATE SOME CONSTANTS -----
!
#ifdef T3D
      fmin = 0.0d0
#endif
      NR = NPM*NRP+NSM*NRAM
#ifdef ROAR_CP
      if(ifqnt.eq.1) nr = nr + nlink
#endif
      N = 3*NR
      BELLY = IBELLY.GT.0
      IER = 0
      NCT = 0
      if (ntc.eq.2 .or. ntc.eq.4) nct = nbonh
      if (ntc.eq.3) nct = nbonh + mbona
      NDFP = N-NCT
      IF(BELLY) NDFP = 3*NATBEL-NCT
      NTNB = 1
      FNQ = SQRT(FLOAT(NDFP))
      IMES = 0
      RMS = 0.0D0
      ERSTOP = .FALSE.
      CONVGD = .FALSE.
      STEEP = .FALSE.
      SKIP = .FALSE.
      NEWSTR = .FALSE.
      NSTCYC = 0
      MSTCYC = KSTCYC
      IF(NTMIN.EQ.2) MSTCYC = MAXCYC
      IF(NTMIN.EQ.1) MSTCYC = NCYC
      IF(NTMIN.GT.0) STEEP = .TRUE.
      FOLD = 0.0D0
      DXST = DX0
      LINMIN = 0
      ITMOUT = 0
      ILMNFL = 0
!
!     ----- PARTITION THE WORKING ARRAY -----
!
      IRSDX = N
      IRSDG = IRSDX+N
      IGINIT = IRSDG+N
      IXOPT = IGINIT+N
      IGOPT = IXOPT+N
!
!     ----- SET SOME PARAMETERS TO BEGIN THE CALCULATION -----
!
      iflag = 0         ! see comments in lbfgs
      m=5           !       id.
      diagco=.false.        !       id.
      xtol = 1.0d-16        !       id.
      iprint(1) = -1        !       id.
      iprint(2) = 0     !       id.
!
      ITERC = 0
      NCALLS = 0
      ITERFM = ITERC
#ifdef ROAR_CP
      ipstep=-1
      if(ncnstr.gt.0)then
        call fixqm(f,x,winv,ierror)
        if(ierror.ne.0)then
           print *,'Call to FIXQM from runmin failed.'
           print *,'Unable to set qm constraints.'
           call mexit(1)
        endif
      endif
```

```
147 #endif
148 !
149 !     ----- LET THE INITIAL SEARCH DIRECTION BE MINUS THE GRADIENT
150 !           VECTOR.  ITERRS GIVES THE ITERATION NUMBER OF THE MOST
151 !           RECENT RESTART , BUT IS SET TO ZERO WHEN STEEPEST DESCENT
152 !           DIRECTION IS USED -----
153 !
154 !=======================================================================
155 !                    (Here is the beginning of a big loop:)
156    20 CONTINUE
157 !=======================================================================
158 !
159 !     ----- GATHER THE SUBMOLECULES INTO THE BOX IF NTB.NE.0 -----
160 !
161       IF (NTB.NE.0) THEN
162          CALL SHIAG(NSPM,NSPSTR,NSP,X,NR,.false.)
163       ENDIF
164       NCALLS = NCALLS+1
165       IF(NCALLS.EQ.NCALLS/NSNB*NSNB) NTNB = 1
166       IF(NTNB.EQ.1 .AND. NCALLS.GT.1) STEEP = .TRUE.
167 !
168 !=======================================================================
169 !
170 !     ----- CALCULATE THE FORCE AND ENERGY -----
171 !
172 !     ----- APPLY BOND CONSTRAINTS IF NECESSARY -----
173 !
174 !=======================================================================
175 !
176 #ifdef ROAR_CP
177       CALL FORCECP(xx,ix,ih,X,FG,ENE(1),VIR,IMES,ifqnt,nquant,
178      $labels,mlabel,nlink,mmqmbo,iqmshk,iqmres,winv)
179 #else
180       CALL FORCE(xx,ix,ih,X,FG,ENE,VIR,IMES)
181 #endif
182       F = ENE(1)
183       NTNB = 0
184       SUM = DDOTP(N,FG,FG)
185       RMS = SQRT(SUM)/FNQ
186 !
187 !     ----- PRINT THE INTERMEDIATE RESULTS -----
188 !
189       if (mod(ncalls,ntpr).eq.0  .or.  ncalls.eq.1) then
190 #ifdef MPI
191         if (master) then
192 #endif
193         call grdmax(n,fg,iatmax,fdmax)
194         iatmax = (iatmax-1)/3 + 1
195         labmax = igraph(iatmax)
196         call opinfo(7)
197         CALL PRINTE(NCALLS,RMS,FDMAX,ENE,iatmax,labmax)
198         close(7)
199 #  ifndef ARCHIVE
200         CALL MINRIT(NRES,X,IGRAPH,LBRES,IPRES,nlink)
201         CALL TIMIT(1,SKIP,6)
202 #ifdef MPI
203         endif
204 #endif
205         IF (SKIP) THEN
206           ITMOUT = 1
207           GO TO 310
208         END IF
209 #  endif
210       end if
211    35 continue
212 !
213 !=======================================================================
214 !
215 !     NTMIN = 3 : LBFGS method (limited memory BFGS)
216 !
217 !=======================================================================
218 !
219       if (ntmin.eq.3) then
```

```
220          do i = 1,n
221             fg(i) = -fg(i)
222          end do
223          call lbfgs(n,5,x,ene(1),fg,.false.,w,iprint,drms,xtol,
224       $                 w(irsdx+1),iflag,fnq,dxm)
225          if (iflag.lt.0) write(*,*)'LBFGS-error',iflag
226          if (iflag.eq.0) go to 300
227          if (iflag.eq.1) then
228             if (NCALLS.ge.maxcyc) then
229                ier=131
230                go to 290
231             else
232                go to 20
233             end if
234          end if
235          write(*,*) 'LBFGS-error',iflag
236          return
237       endif
238
239 !
240 !=====================================================================
241 !
242 !     ----- DO SOME STEEPEST STEPS BEFORE ENTERING THE CONJUGATE
243 !           GRADIENT METHOD -----
244 !
245 !=====================================================================
246 !
247       IF (STEEP) then
248         NSTCYC = NSTCYC+1
249         IF (NSTCYC.le.MSTCYC) then
250 !
251           IF(DXST.LE.CRITS) DXST = DXSTM
252           DXST = DXST/2.0D0
253           IF(F.LT.FOLD) DXST = DXST*2.4D0
254           DXSTH = DXST/SQRT(SUM)
255           IF(NSTCYC.le.1 .or. F.le.FMIN) then
256             FMIN = F
257             NFOPT = NCALLS
258             DO 40 I = 1,N
259               W(IXOPT+I) = X(I)
260               W(IGOPT+I) = -FG(I)
261    40         CONTINUE
262           end if
263 !
264 !     ----- CHECK FOR CONVERGENCE -----
265 !
266           IF(RMS.LE.DRMS) GO TO 300
267           IF(NCALLS.ge.MAXCYC) then
268             IER = 131
269             GO TO 290
270           end if
271           FOLD = F
272           DO 50 I = 1,N
273             X(I) = X(I)+DXSTH*FG(I)
274    50     CONTINUE
275 #ifdef ROAR_CP
276       if(ncnstr.gt.0)then
277         call fixqm(w(ixopt+1),x,winv,ierror)
278         if(ierror.ne.0)then
279            print *,'Call to FIXQM from runmin failed.'
280            print *,'Unable to set qm constraints.'
281            call mexit(1)
282         endif
283       endif
284 #endif
285           GO TO 20
286 !
287         ELSE
288 !                               (arrive here when finished with this
289 !                                set of steepest descent cycles)
290           STEEP = .FALSE.
291           NEWSTR = .TRUE.
292           NSTCYC = 0
```

```
293           MSTCYC = KSTCYC
294         END IF
295       END IF
296 !
297 !=======================================================================
298 !
299 !     ----- START OF CONJUGATE GRADIENT STEPS -----
300 !
301 !=======================================================================
302 !
303       DO 60 I = 1,N
304         FG(I) = -FG(I)
305    60 CONTINUE
306 !
307       IF (.not. NEWSTR .and. NCALLS.ge.2) go to 82
308    70   DO 80 I=1,N
309           W(I) = -FG(I)
310    80   CONTINUE
311         ITERRS = 0
312         IF(NEWSTR) ITERC = 0
313         IF(ITERC.GT.0) GO TO 140
314    82 CONTINUE
315 !
316       GNEW = DDOTP(N,W,FG)
317       IF(NEWSTR .OR. NCALLS.EQ.1) GO TO 100
318       FCH = F-FMIN
319 !
320 !     ----- STORE THE VALUES OF X, F AND G, IF THEY ARE THE BEST THAT
321 !           HAVE BEEN CALCULATED SO FAR. TEST FOR CONVERGENCE ----
322 !
323       IF(FCH) 100,90,130
324    90 IF(GNEW/GMIN.LT.-1.0D0) GO TO 120
325   100 FMIN = F
326       GSQRD = SUM
327       NFOPT = NCALLS
328       DO 110 I=1,N
329         W(IXOPT+I) = X(I)
330         W(IGOPT+I) = FG(I)
331   110 CONTINUE
332   120 IF(RMS.LE.DRMS) GO TO 300
333 !
334 !     ----- TEST IF THE VALUE OF MAXCYC ALLOWS ANOTHER CALL OF FUNCT ---
335 !
336   130 IF(NCALLS.ge.MAXCYC) then
337         IER = 131
338         GO TO 290
339       end if
340       IF(.NOT.NEWSTR .AND. NCALLS.GT.1) GO TO 180
341 !
342 !    ------ This section is executed at the beginning of a conjugate
343 !           gradient set of minimization steps.
344 !
345 !     ----- SET DFPR TO DXO*GSQRD. DFPR IS THE REDUCTION IN THE FUNCTION
346 !           VALUE. STMIN IS USUALLY THE STEP-LENGTH OF THE MOST RECENT
347 !           LINE SEARCH THAT GIVES THE LEAST VALUE OF F -----
348 !
349 !  --- dac change, 10/91:  return to original idea of trying to
350 !       go downhill by the absolute amount, DFPRED (which defaults
351 !       to 1 kcal/mol, see data statement above).  This can eliminate
352 !       very bad initial conjugate gradient steps.
353 !
354 !  ***amber 4.0 code:
355 !
356 !     DFPR = DXST*GSQRD
357 !     STMIN = DXST
358 !
359       DFPR = DFPRED
360       STMIN = DFPRED/GSQRD
361 !
362       NEWSTR = .FALSE.
363 !
364 !=======================================================================
365 !
```

```
366 !     ----- Begin the main conguate gradient iteration -----
367 !
368 !=================================================================
369 !
370   140 ITERC = ITERC+1
371 !
372       FINIT = F
373       GINIT = 0.0D0
374       DO 150 I=1,N
375         W(IGINIT+I) = FG(I)
376   150 CONTINUE
377       GINIT = DDOTP(N,W,FG)
378       IF(GINIT.GE.0.0D0) GO TO 260
379       GMIN = GINIT
380       SBOUND = -1.0D0
381       NFBEG = NCALLS
382       IRETRY = -1
383 !
384       STEPCH = MIN(STMIN,ABS(DFPR/GINIT))
385       STMIN = DXSTM
386 !
387   160 STEP = STMIN+STEPCH
388       DXST = STEP
389       sWORK = 0.0D0
390       DO 170 I=1,N
391         X(I) = W(IXOPT+I)+STEPCH*W(I)
392         sWORK = MAX(sWORK,ABS(X(I)-W(IXOPT+I)))
393   170 CONTINUE
394 #ifdef ROAR_CP
395       if(ncnstr.gt.0)then
396         call fixqm(w(ixopt+1),x,winv,ierror)
397         if(ierror.ne.0)then
398            print *,'Call to FIXQM from runmin failed.'
399            print *,'Unable to set qm constraints.'
400            call mexit(1)
401         endif
402       endif
403 #endif
404       IF(sWORK.GT.0.0D0) GO TO 20
405 !     "work = swork" may not be needed - wont hurt.  -gls
406       work = swork
407 !
408 !     ----- TERMINATE THE LINE SEARCH IF STEPCH IS EFFECTIVELY ZERO ---
409 !
410       IF(NCALLS.GT.NFBEG+1 .or. ABS(GMIN/GINIT) .gt. 0.2D0) then
411 #ifdef MPI
412         if (master) WRITE(6,370)
413 #else
414         WRITE(6,370)
415 #endif
416         STEEP = .TRUE.
417         LINMIN = LINMIN+1
418       end if
419       go to 270
420 !
421   180 WORK = (FCH+FCH)/STEPCH-GNEW-GMIN
422       DDSPLN = (GNEW-GMIN)/STEPCH
423       IF (NCALLS.GT.NFOPT) then
424         SBOUND = STEP
425       ELSE
426         IF(GMIN*GNEW.LE.0.0D0) SBOUND = STMIN
427         STMIN = STEP
428         GMIN = GNEW
429         STEPCH = -STEPCH
430       END IF
431       IF(FCH.NE.0.0D0) DDSPLN = DDSPLN+(WORK+WORK)/STEPCH
432 !
433 !     ----- TEST FOR CONVERGENCE OF THE LINE SEARCH, BUT FORCE ATLEAST
434 !           TWO STEPS TO BE TAKEN IN ORDER NOT TO LOSE QUADRATIC
435 !           TERMINATION -----
436 !
437       IF(GMIN.EQ.0.0D0) GO TO 270
438       IF(NCALLS.LE.NFBEG+1) GO TO 200
```

```
439       IF(ABS(GMIN/GINIT).LE.0.2D0) GO TO 270
440 !
441 !     ----- APPLY THE TEST THAT DEPENDS ON THE PARAMETER MAXLIN -----
442 !
443   190 IF(NCALLS.LT.NFOPT+MAXLIN) GO TO 200
444 !
445 !     ----- POSSIBLE NON BONDED UPDATE.  MAKE A RESTART -----
446 !
447 #ifdef MPI
448       if (master) WRITE(6,370)
449 #else
450       WRITE(6,370)
451 #endif
452       STEEP = .TRUE.
453       LINMIN = LINMIN+1
454       GO TO 270
455 !
456   200 STEPCH = 0.5D0*(SBOUND-STMIN)
457       IF(SBOUND.LT.-0.5D0) STEPCH = 9.0D0*STMIN
458       GSPLN = GMIN+STEPCH*DDSPLN
459       IF(GMIN*GSPLN.LT.0.0D0) STEPCH = STEPCH*GMIN/(GMIN-GSPLN)
460       GO TO 160
461 !
462 !     ----- CALCULATE THE VALUE OF BETA IN THE NEW DIRECTION -----
463 !
464   210 SUM = DDOTP(N,FG,W(IGINIT+1))
465       BETA = (GSQRD-SUM)/(GMIN-GINIT)
466 !
467 !     ----- TEST THAT THE NEW SEARCH DIRECTION CAN BE MADE DOWNHILL.
468 !           IF NOT THEN TRY TO IMPROVE THE ACCURACY OF THE LINE
469 !           SEARCH -----
470 !
471       IF(ABS(BETA*GMIN).LE.0.2D0*GSQRD) GO TO 220
472       IRETRY = IRETRY+1
473       IF(IRETRY.LE.0) GO TO 190
474 !
475   220 IF (F.LT.FINIT) ITERFM = ITERC
476       IF (ITERC.ge.ITERFM+MXFCON) then
477 #ifdef MPI
478         if (master) WRITE(6,370)
479 #else
480         WRITE(6,370)
481 #endif
482         STEEP = .TRUE.
483         LINMIN = LINMIN+1
484         GO TO 270
485       END IF
486       DFPR = STMIN*GINIT
487 !
488 !     ----- BRANCH IF A RESTART PROCEDURE IS REQUIRED DUE TO THE
489 !           ITERATION NUMBER OR DUE TO THE SCALAR PRODUCT OF
490 !           CONSECUTIVE GRADIENTS -----
491 !
492       IF(IRETRY.GT.0) GO TO 70
493       IF(ITERRS.EQ.0) GO TO 240
494       IF(ITERC-ITERRS.GE.N) GO TO 240
495       IF(ABS(SUM).GE.0.2D0*GSQRD) GO TO 240
496 !
497 !     ----- CALCULATE GAMA IN THE NEW SEARCH DIRECTION.  GAMDEN IS
498 !           SET BY THE RESTART PROCEDURE -----
499 !
500       GAMA = DDOTP(N,FG,W(IRSDG+1))
501       SUM  = DDOTP(N,FG,W(IRSDX+1))
502       GAMA = GAMA/GAMDEN
503 !
504 !     ----- RESTART IF THE NEW SEARCH DIRECTION IS NOT SUFFICIENTLY
505 !           DOWNHILL ----
506 !
507       IF(ABS(BETA*GMIN+GAMA*SUM).GE.0.2D0*GSQRD) GO TO 240
508 !
509 !     ----- CALCULATE THE NEW SEARCH DIRECTION -----
510 !
511       DO 230 I=1,N
```

```
512         W(I) = -FG(I)+BETA*W(I)+GAMA*W(IRSDX+I)
513   230 CONTINUE
514 !
515 !     --- cycle back for more conjugate gradient steps:
516 !
517       GO TO 140
518 !
519 !     ----- APPLY THE RESTART PROCEDURE -----
520 !
521   240 GAMDEN = GMIN-GINIT
522       DO 250 I=1,N
523         W(IRSDX+I) = W(I)
524         W(IRSDG+I) = FG(I)-W(IGINIT+I)
525         W(I) = -FG(I)+BETA*W(I)
526   250 CONTINUE
527       ITERRS = ITERC
528       GO TO 140
529 !
530 !     ----- SET IER TO INDICATE THAT THE SEARCH DIRECTION IS UPHILL ---
531 !
532   260 CONTINUE
533       STEEP = .TRUE.
534 #ifdef MPI
535       if (master) WRITE(6,370)
536 #else
537       WRITE(6,370)
538 #endif
539       LINMIN = LINMIN+1
540 !
541 !     ----- ENSURE THAT F, X AND G ARE OPTIMAL -----
542 !
543   270 IF(NCALLS.ne.NFOPT) then
544         F = FMIN
545         DO 280 I=1,N
546           X(I) = W(IXOPT+I)
547           FG(I) = W(IGOPT+I)
548   280   CONTINUE
549       end if
550       IF(LINMIN.GT.4) THEN
551          ILMNFL = 1
552          GO TO 310
553       END IF
554       IF(STEEP) GO TO 20
555       IF(IER.EQ.0) GO TO 210
556   290 CONTINUE
557 #ifdef MPI
558       if (master) then
559 #endif
560       IF(IER.EQ.129) WRITE(6,320)
561       IF(IER.EQ.130) WRITE(6,330)
562       IF(IER.EQ.131) WRITE(6,340)
563       IF(IER.EQ.132) WRITE(6,350)
564 #ifdef MPI
565       endif
566 #endif
567       ERSTOP = .TRUE.
568       GO TO 310
569 !
570   300 CONTINUE
571       CONVGD = .TRUE.
572 !
573 !     ----- WRITE THE FINAL RESULTS -----
574 !
575   310 CONTINUE
576 #ifdef MPI
577       if (master) then
578 #endif
579       WRITE(6,380)
580       call grdmax(n,fg,iatmax,fdmax)
581       iatmax = (iatmax-1)/3 + 1
582       labmax = igraph(iatmax)
583       call opinfo(7)
584       CALL PRINTE(NCALLS,RMS,FDMAX,ENE,iatmax,labmax)
```

```
585       close(7)
586       IF (ITMOUT.EQ.1) WRITE(6,360)
587       IF (ILMNFL.EQ.1) WRITE(6,390)
588 #ifdef MPI
589       endif
590 #endif
591       RETURN
592 !
593   320 FORMAT('  LINE SEARCH ABANDONED ... PROBLEM WITH G')
594   330 FORMAT('  SEARCH DIRECTION IS UPHILL ')
595   340 FORMAT('  MAXIMUM NUMBER OF F EVALUATION EXCEEDED')
596   350 FORMAT('  THE VALUE OF F COULD NOT BE REDUCED')
597   360 FORMAT(/4X,'WARNING ... TIME LIMIT EXCEEDED ... TO BE RESTARTED')
598   370 FORMAT(/4X,' .... RESTARTED DUE TO LINMIN FAILURE ...')
599   380 FORMAT(/ /20X,'FINAL RESULTS',/)
600   390 FORMAT(/5X,'***** REPEATED LINMIN FAILURE *****')
601 !
602       END
603
```