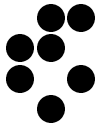
Institut „Jožef Stefan", Ljubljana, Slovenija



IJS delovno poročilo
DP-8100

CFHHM CODE PARALLELIZATION ON SMP MACHINES 1995-1999

R. Krivec
Department of Theoretical Physics

CFHHM CODE PARALLELIZATION ON SMP MACHINES 1995-1999

R. Krivec,
Department of Theoretical Physics,
J. Stefan Institute,
POB 3000, 1000 Ljubljana, Slovenia

99/08/02

ABSTRACT

Parallelization of a high-precision physics code for solving coupled systems of differential equations with piecewise matrix expansions is presented. The transition from vector-type code to parallel code is accomplished by partially inverting a loop nest. The implementation is by compiler directives on an SMP machine. Because of piecewise algorithm, the parallel code is slower than vector code on a single thread, but overtakes it with more than 4 processors used. The presented example is extremely elegant because only a small piece of code is changed, and no redundant calculations are introduced.

SUMMARY

This report presents a case of code parallelization using mathematical insight rather than MPI or redundant distributed calculations, originally performed in 1995. The starting point is code with vector and superscalar efficiency in single-threaded mode of up ot 80%.

The presented measurements are for the He atom ground state (the simplest CFHHM example), in which matrix operations use up to 70% of CPU time. Because of matrix recursion formulas, single matrix operations cannot be efficiently parallelized.

Part I of the report presents a description of CFHHM. Part II reports the parallelization done in 1998 on a SGI Power Challenge. Part III reports the first parallelization done in 1995 on a SPP-1000. The basic code rewriting was done in 1995 for the SPP-1000. However, this machine had its problems, so the releveant results are the Power Challenge ones. The basic trend is the same in both cases.

The result is that on a single processor, the modified and parallelized code runs 2-3 times slower than the original code. To reach the same speed, 4 processors are necessary.

The main reason that the parallelized code runs 2-3 times slower in single-threaded mode is not redundant calculations, but the fact that in the calculations the recursion process must be repeated on many subintervals, between which some matrices must be rejoined and inverted. If only one such interval could be used, parallelization would be almost perfect.

FILES

The information in this report is extracted form the following files: sgi98_cfhhm_mp_preliminary.txt, bench_bm_cfhh1.txt, cfhhoptsppp.txt.

PURPOSE

(1) To convert a program with predominantly matrix operations, and optimized for vector and superscalar machines, to parallel using coarse-grained parallelization obtained with explicit code rearrangement, based on mathematical insight, possibly without introducing repetition of intermediate calculations.
(2) Find the most suitable program loop topology on different machines, and possibly a topology efficient on several machines.

PART I

DESCRIPTION

CFHHM (Correlation Function Hyperspherical Harmonic Method) originated and developed by V.B. Mandelzweig (Hebrew University, Jerusalem), M.I. Haftel (Naval Research Laboratory, Washington) and R. Krivec (J. Stefan Institute, Ljubljana) is a method to calculate very precise values of observables in the quantum nonrelativistic three body problem, especially in atomic physics.

Selected results:
- The best value of annihilation rate of positronium negative ion [1]
- Sticking probability and fusion rate for the muon-deuteron-triton mesomolecule in the muon-catalyzed fusion process; the results pointed out several inconsistencies in the literature [2]
- Resonances in the doubly excited He atom [3], showing that in CFHHM, in contrast to the variational calculations, no stabilization is needed [3]
- The best calculation of hyperfine splitting in the muonic Helium atom, and discovery of wrong convergence of variational calculations [4,5,6]

The main programs in CFHHM are M2MK and EWF2P. M2MK calculates matrix elements and EWF2P solves a system of coupled ordinary differential equations in one variable (hyperradius), which represent the three-body Schroedinger equation. At the origin, the exact analytic form including logarithmic terms is represented as matrix recursion relations. On the rest of the interval, the solution is expressed as matrix Taylor series on small subintervals, calculated via matrix recursion relations, and joined at the boundaries [7].

M2MK

Calculates matrix elements (runs once, has negligible CPU time in the complete CFHHM calculation).
m2mk is a mixed integer/REAL*8 program, written for clarity, and not optimized. MFLOPS is meaningless for it. Its speed is presented as an example of how catastrophically low efficiency such "nice" programs can have, especially on superscalar machines.

Dimension declarations:
NDMU = 33, NDP0 = 104, NDP = 102,
NDGO = 32, NNQSI = 8.
Km = 2*MUU (input), where Km is the maximum global angular momentum.

This program is not parallelized.

EWF2P

Solves the coupled system of differential equations in one variable [7] by prescribing the energy E, propagating the solution from two sides towards a matching hyperradius (ZMATCH), calculating a "joining determinant" D(E), and iterating E until D(E) = 0. Typically 7 iterations with fixed values of E are necessary to find the zero to more than 10 significant digits, which assures the wave function accuracy of at least 6-7 significant digits.

The CPU times quoted are for one iteration (complete integration with fixed energy parameter).

ewf2p is a "vector" floating-point (REAL*8) program.

Dimension declarations:
NDMU variable, ISYM = 0, NDP = 102, NDP0 = 104, NDCN = 22 (size is 790 MB at NDMU = 33).
NSU is the number of coupled equations (matrix size) in the recursive calculation of a matrix Taylor expansion of the solution on sub-intervals.
NDMU is the main dimension and is varied to see its effect at small NSU.
The relations are NDMU >= Km / 2, NSU = const * Km**2.

CPU time model:
in the first approximation we count only the main part of RRP* (recursion relations, see below):

```
            T = C(IPWUU) * NSU**3 * Nint * (JMAX**2)/2 + T0
```

where C is a constant, NSU is the number of coupled equations, Nint is the number of hyperradial intervals, JMAX is the number of powers in the matrix Taylor expansion of the solution, and T0 is the time spent on the first interval (small for He atom). C depends on IPWUU because the power expansion of the effective potential introduces additional summations whose CPU time is proportional to NSU**2 * IPWUU * JMAX, and changes the geometry of the most important loops.

Nint can be approximated as follows. Interval boundaries are determined by Z(n) = min((1 + TOL)**n * ZMATCH, DZU), whence we get that the lengths Z(n+1) - Z(n) increase up to Z(n1) = DZU / TOL, and n1 = (1 / TOL) * ln [DZU / (TOL * ZMATCH)]. Then if Z(n1) < ZTOP, Nint = n1 + (ZTOP - Z(n1)) / DZU.
The actual Nint values were calculated by a trivial program (aux700.f), because the CFHHM program does not enumerate them.

Recursion formulas evaluation:

ewf2p subroutine ddca calls several variants of rrp* subroutines which evaluate the recursion formulas with different loop topologies, and with and without calls to DAXPY and DGEMM. Some rrp* routines are also completely reordered for parallelization. The list of the most important versions is as follows (index M1 =< JMAX runs over Taylor terms in the solution, index IPX =< IPWUU runs over powers in the effective potential):

Name	Originating rrpsm version	IRRPV (ddca)	Short description
rrp11	1.1	1	Single M1 loop outside, IPX inside
rrp30	3.0	30	as rrp11, eliminated power evaluation
rrp30a	3.0V	301	as rrp30, but calls DGEMM
rrp30b	3.0V1	302	as rrp30a, but calls DAXPY
rrp31c	3.1.1V1	314	M1 loop distributed and moved inside
rrp31e		316	as rrp31c, but DO-loops only

References

[1] R. Krivec and V.B. Mandelzweig, Phys. Rev. A47, 911 (1993).
[2] R. Krivec and V.B. Mandelzweig, Phys. Rev. A52, 221 (1995).
[3] S. Berkovic, R. Krivec, V.B. Mandelzweig and L. Stotland, Phys. Rev. A55, 988 (1997).
[4] R. Krivec and V.B. Mandelzweig, Phys. Rev. A56, 3614 (1997).
[5] R. Krivec and V.B. Mandelzweig, Phys. Rev. A57, 4976 (1998).
[6] R. Krivec and V.B. Mandelzweig, Proc. XVI-th European Few-Body Conference, Autrans, June 1-6, 1998.
[7] R. Krivec, M.I. Haftel and V.B. Mandelzweig, Journal of Computational Physics 123, 149 (1996).

PART II

CFHHM CODE PARALLELIZATION
SGI POWER CHALLENGE
98/11/19

For this work the starting point are the routine versions written in 1995 (Part III).

PROBLEM

CFHHM He atom ground state energy search. We solve a NxN coupled system of ordinary differential equations in one variable on about 100 intervals of the variable. On each interval, the system is propagated by expanding the solution in matrix Taylor series. This results in matrix recurrence formulas in which the column index (I) is memory-contiguous, and columns of the solution matrix are independent.
As explained below, most of the CPU time is consumed using linear algebra either by DO loops or by Blas routines. Therefore the speed as measured by the total program CPU time is a good measure of the level of optimization of the central part of the code.

PROGRAM STRUCTURE

EWF2P calls DDCA which in turn calls RRP* where recursion formulas are evaluated. There are two basic possibilites: RRP31C and RRP41C. (Other possibilities are less optimized single-threaded verions.)

RRP31C calculates recursions by having the index I in several innermost loops (vector style). The central part of the program is as follows:

```
           DO  100  N1 = 3,JMAX1
      C         RESET  CNT  (CNT(.,.,1), CNT(.,.,2) ARE INPUT PARAMETERS).
                DO  62  J = 1,MM
                    DO  60  I = 1,MM
                        CNT(I,J,N1) = ZERO
         60         CONTINUE
         62     CONTINUE
                JP = N1 - 1
      C         SUM OVER R (R = -1,..., JP-2).
                IF (IRTC0 .EQ. 0) THEN
      C             FIRST LOOP: EXCLUDE CNT(.,.,1). THIS ASSUMES THAT RT
      C             WAS CALCULATED ON THE PREVIOUS INTERVAL,
      C             AND THAT CNT(.,.,1) = I.
                    JP1 = JP - 1
                ELSE
      C             FIRST LOOP: DO NOT EXCLUDE CNT(.,.,1) (DO ALL).
                    JP1 = JP
                ENDIF
                DO  74  M1 = 1,JP1
                    JR    = M1 - 2
                    ZZZ72 = JP - JR - 2
                    DO  72  K = 1,MM
                        DO  70  J = 1,MM
                            RR(J,K) = ZZZ72 * PP(J,K,M1) + QQ(J,K,M1)
         70             CONTINUE
         72         CONTINUE
                    CALL DGEMM('N', 'T', MM, MM, MM,
           &                   ONE, CNT(1,1,N1-JR-2), NCDS,
           &                   RR, NCDS,
           &                   ONE, CNT(1,1,N1), NCDS)
         74     CONTINUE
                IF (IRTC0 .EQ. 0) THEN
      C             SECOND LOOP: CNT(1) = I (IMPLICIT). SEE ABOVE.
                    M1 = JP
                    DO  82  J = 1,MM
                        DO  80  I = 1,MM
                            CNT(I,J,N1) = CNT(I,J,N1) + QQ(J,I,M1)
         80             CONTINUE
         82         CONTINUE
                ENDIF
      C         FINISH  C(JP) .
                DO  92  J = 1,MM
                    DO  90  I = 1,MM
                        CNT(I,J,N1) = ZZZ71 * CNT(I,J,N1)
         90         CONTINUE
         92     CONTINUE
        100 CONTINUE
```

RRP41C calculates recursions by having the index I in a single outermost loop, to exploit independence of matrix columns for the parallelization of the code. RRP41C is executed once per interval, therefore threads are generated and rejoined on each interval. The inner loops are

```
C$DOACROSS LOCAL(CNTX, CNTY, N1, J, JP, JP1, JR, ZZZ71)
      DO  122  I = 1,MM
C         INITIALIZE CNTX(,,2); CNTX(,,1) NOT NEEDED.
          DO  70  J = 1,MM
              CNTX(J,2) = CNT(I,J,2)
   70     CONTINUE
C         THE WEIGHTED COPY.
          DO  71  J = 1,MM
              CNTY(J,2) = CNTX(J,2)
   71     CONTINUE
          DO  120  N1 = 3,JMAX1
              JP = N1 - 1
C             RESET CURRENT CNTX.
              DO  72  J = 1,MM
                  CNTX(J,N1) = ZERO
   72         CONTINUE
C             SUM OVER R (R = -1,..., JP-2).
C             FIRST LOOP: EXCLUDE CNT(,,1).
              JP1 = JP - 1
              DO  90  M1 = 1,JP1
                  JR = M1 - 2
                  CALL DGEMV('N', MM, MM, ONE, PP(1,1,M1), NCDS,
     &                       CNTY(1,N1-JR-2), 1,
     &                       ONE, CNTX(1,N1), 1)
                  CALL DGEMV('N', MM, MM, ONE, QQ(1,1,M1), NCDS,
     &                       CNTX(1,N1-JR-2), 1,
     &                       ONE, CNTX(1,N1), 1)
   90         CONTINUE
C             SECOND LOOP: CNT(,,1) = I (IMPLICIT).
              M1 = JP
              DO  92  J = 1,MM
                  CNTX(J,N1) = CNTX(J,N1) + QQ(J,I,M1)
   92         CONTINUE
C             FINISH  CNT(JP) .
              ZZZ71 = - ONE / DBLE(JP * (JP - 1))
              DO  100  J = 1,MM
                  CNTX(J,N1) = ZZZ71 * CNTX(J,N1)
  100         CONTINUE
C             THE WEIGHTED COPY.
              DO  102  J = 1,MM
                  CNTY(J,N1) = (N1 - 1) * CNTX(J,N1)
  102         CONTINUE
C             STORE INTO CNT, LEAVE CNTX FOR ITERATION.
              DO  110  J = 1,MM
                  CNT(I,J,N1) = CNTX(J,N1)
  110         CONTINUE
  120     CONTINUE
  122 CONTINUE
```

PROGRAM PROFILES

Profiles were generated using pixie. It was checked that a single pass (DDCA call) gives the same percentages as a full run (about 7 calls).

Single threaded mode:

Block	N	Percentage of total EWF2P CPU time
COMPLIB_DGEMM_HOISTC *1	49	79.7
RRP31C		90.9 *2
(Other)		9
COMPLIB_DGEMM_HOISTC *1	121	88.0
COMPLIB_DGEMM_HOISTC *1	225	

*1 /usr/lib64/libblas.so:/vince/6.2-mar09/work/cmplrs/libblas/complib_dgemm_hoistc.f
*2 ewf2p.e__usr_bin_f77_r8000_mips4_64_O3_lcomplib.sgimath:rrp31c.f; cumulative percentage.

Multi threaded mode:

Block	N	Threads	Percentage of total EWF2P CPU time
DGEMV_MV *1	49	4	82.9
RRP41C *2			99.7
DGEMV_MV *1	121	6	82.9 (?)

*1 /usr/lib64/libblas.so:/vince/6.2-mar09/work/cmplrs/libblas/dgemv.f
*2 ewf2p.e__usr_bin_f77_r8000_mips4_64_O3_mp_lcomplib.sgimath: rrp41c_mp.f; cumulative precentage.

RUNS

Single threaded:
RRP31C (IRRPV = 314)
R8000 75 or 90 MHz
/usr/bin/f77 -r8000 -mips4 -64 ... -lcomplib.sgimath

Multi threaded:
RRP41C (IRRPV = 414) (c$doacross),
R8000 mixed frequency (2 x 90 MHz, 4 x 75 MHz)
/usr/bin/f77 -r8000 -mips4 -64 -mp ... -lcomplib.sgimath
/usr/bin/f77 -r8000 -mips4 -64 -mp ... -lcomplib.sgimath_mp

(The latter seem both equivalent; any differences would only be felt in minor parts of program. But to measure the hand optimization speed, it is better to link the non-parallel complib.sgimath.)

RESULTS

Table II.1. Km = 24, N = 49. 75 MHz.

RRP*** version	Threads	Library linked explicitly	CPU time per determinant*	Speedup RRP31C	Speedup RRP41C
RRP31C (vectorized)	1	complib.sgimath	10	1	
RRP41C (parallelized)	1	complib.sgimath	24	0.4	1
	2		14	0.7	1.7
	4		10	1	2.4
	6		8.5	1.2	2.8
RRP41C (parallelized)	1	complib.sgimath_mp			
	2		16		
	4		10		
	6		8.1		

* One DDCA call (all intervals). There are about 7 calls per run. The CPU time is the average of all calls in a run. It was verified that the run for a single pass gives the CPU time which is very close to this averaged time. 80% of the CPU time is used by DGEMM.

Table II.2. As Table II.1, but Km = 40, N = 121. 75 MHz.

RRP*** version	Threads	Library linked explicitly	CPU time per determinant*	Speedup RRP31C	Speedup RRP41C
RRP31C (vectorized)	1	complib.sgimath	111	1	
RRP41C (parallelized)	1	complib.sgimath			
	2				
	4				
	6		148		

APPENDIX II.A. CONTROL FILES

(1) M2MK control files:

```
------------------------------------------------------------------------------
98/11/19  f77
T                          LSTORE (UNFORMATTED OUTPUT)
   1                       INUSEL (0: UNSYMMETRIC; 1: SYMMETRIC)
TT                         PCFC, PVC (PRINT F, V POWER EXPANSIONS)
                 1.0                1.0                -1.0  M1/2/3 (-1: INF)
  20   0   F                    MUU, IPOS, PSYM
   1   1  -1  -1   0               IPFL, IPFU, IPVL, IPVU, IPWU
   1                                                        NFPAR
   1                -2.0               -2.0                1.0  J, FPAR(.,J)
   2                                   MODF (INTERPRETATION OF FPAR BY CFC...)
   1                                                        NVPAR
   1                -2.0               -2.0                1.0  J, VPAR(.,J)
   0                                   MODV (INTERPRETATION OF VPAR BY VC...)
  16  16                                      (NGOA(I), I = 1,2)
   4   4                                      (NSUBA(I), I = 1,2)
  16  16  16  16  16  16  16  16  16  16      (NGOL(I,1), I = 1,10)
   4   4   4   4   4   4   4   4   4   4      (NSUBL(I,1), I = 1,10)
  16  16  16  16  16  16  16  16  16  16      (NGOL(I,2), I = 1,10)
   4   4   4   4   4   4   4   4   4   4      (NSUBL(I,2), I = 1,10)
   0.100   0.030                          DAS, DLS
TT                 POMEGA, POTHER (WORKING RUNS: TT)
TFFFF              PCON, PTEST, PTEST1, PRES (WORKING RUNS: TFFF), PCONTR
FFTFFFF            PT0O, PT1O, PRESO, PTG, PTGA, PTGL, PTJ (WRKNG RUNS: F...F)
   4            NSPR (MAX. NO. ROWS/COLUMS OF MATRICES TO BE PRINTED)
   0    itestq
------------------------------------------------------------------------------
#/bin/csh -f
# m2mk-

set pro = m2mk
set do = "~/sgi98/out"
set ds = "/dsk3/tmp/krivec"

set fi = "1a0"
set fo = "1aa4001"

set eex = "56p120a6432a_773"
set zzzexe = "~/cfhh/pd/${pro}.e${eex}"
limit stacksize 500000
set nicev = 10

source M2MK0000.csh
------------------------------------------------------------------------------
```

(2) EWF2P control files (example):

```
--------------------------------------------------------------------------------
98/11/19  f77 Km = 24 test
 314            IRRPV (314: 3.1.1V1)
FFFFFFFFF       LTDD,LTDDR,LTVICD,LTRRPS,LPRBIN,LPRZ0M,LTCPU,LTCRU,LPRPQ
FFFFFF          LPRCV0, LPRXX, LPRCV, *LPRCVM, LPRMAT, LPRWF
TFT             LTZE, LTZEIF, LXLAST (IPROG = 200)
T               LTDET
FF              LDOUBL, LDOUBM
   8            IINTPR
  20 200        LM, IPROG
   0            IENXL (LMOD = 5, 6 ONLY)
   0            I0D
   0            IRTC0
   0  49   1                              IPWUU, NSU, INUSEL
  20  16                                  NM, JMAX
    0.50    0.50  9999.0                  BC0, BC1, RHOUW
2.8000          3.0000                    A, B
  +1.0000000D-14   0.1000000D+00  5.00000000D+00    EPSP, TOL, DZU
    0.50    5.00  120.00                  ZM, ZMATCH, ZTOP
   0                                      NHH
   1                                      LMOD
2.9                         3.00            1.00    ENXL, ENXU, ENXS
   1                                      IPIO (0: M1M-, 1: M2M-)
    0.00  999.00   1   1                  Z0MTTT, Z0MTTD, IPRTTT, IPCTTT
--------------------------------------------------------------------------------
#!/bin/csh -f

setenv MP_SET_NUMTHREADS 1

set pro = "ewf2p"
set pr1 = "m2mk"
set di = "~/sgi98/inp"
set do = "~/sgi98/out"
set ds = "/dsk3/tmp/krivec"

# All blas/linpack routines from system libraries.
set eex = "_usr_bin_f77_r8000_mips4_64_O3_mp_lcomplib.sgimath"

set km = "6"

set fm = "1a64001"
set fi = "1a${km}0"

set irr = cat $pro$fi.i|grep IRRPV|awk '{print $1}'
set fo = "0${km}cfeci555b33a"
set fo1 = "${pro}${fm}${fo}_${eex}_${irr}_t${MP_SET_NUMTHREADS}"
set foo1 = ""

set zzzexe = "~/sgi98/pd/${pro}.e_${eex}"
limit stacksize 300000
set nicev = 0

set sigrun = "${do}/${pr1}${fm}.sig"    # start signal file

source EWF2P0000.csh

\rm -f $FLU30 $FLU40
--------------------------------------------------------------------------------
```

APPENDIX II.B. OUTPUT FILES

These runs were done without pixie. Multi-pass runs are listed; the purpose was to check that the zero is found to the same precision in all cases.

Km = 24, N = 49:

```
--------------------------------------------------------------------------------
75 MHz:
ewf2p1a6400106cfeci555b33a__usr_bin_f77_r8000_mips4_64_O3_lcomplib.sgimath_314.2
 -RRLSDE- 1.1, 96/08/02. 1.0, 93/05/18. 1.4, 91/03/13 (VICDF)
 -RRP31C- 2.0, 96/08/05. -1.0; -RRPSM 3.1.1V1. -3.1V1S -3.0V1S -3.0VS
 DET:     -7.6262076063431188 10** 37  ENERGY      2.7999999999999998
 DET:      6.5711433071813374 10** 37  ENERGY      2.9999999999999996
 DET:      2.0077284429255133 10** 36  ENERGY      2.9065738535396579
 DET:      3.7022999515154980 10** 33  ENERGY      2.9037282905579587
 DET:     -1.7360133514077343 10** 29  ENERGY      2.9037230387903326
 DET:     -3.8760312384817515 10** 23  ENERGY      2.9037230390365769
 DET:      1.9216012805935490 10** 23  ENERGY      2.9037230390365778
    1  0.1921601280593552E-13  H   0.0000E+00      2.9037230390365778
    5 + 2 DET. EVALUATIONS OF       10.11 SEC.
   USER       70.79     DIF       70.79     REF        0.00     SYS        0.44

--------------------------------------------------------------------------------
ewf2p1a6400106cfeci555b33a__usr_bin_f77_r8000_mips4_64_O3_mp_lcomplib.sgimath_41
 -RRLSDE- 1.1, 96/08/02. 1.0, 93/05/18. 1.4, 91/03/13 (VICDF)
 -RRP41C-1.0 98/11/09 RRPSM 4.1.1V1S 96/02/09 -4.1PV1S 4.1PV1 -M_4.1P
 DET:     -7.6262076063431072 10** 37  ENERGY      2.7999999999999998
 DET:      6.5711433071813330 10** 37  ENERGY      2.9999999999999996
 DET:      2.0077284429255240 10** 36  ENERGY      2.9065738535396579
 DET:      3.7022999514429218 10** 33  ENERGY      2.9037282905579587
 DET:     -1.7360127395319718 10** 29  ENERGY      2.9037230387903326
 DET:     -3.8867156868107071 10** 23  ENERGY      2.9037230390365769
 DET:      2.0731391735528364 10** 23  ENERGY      2.9037230390365778
    1  0.2073139173552839E-13  H   0.0000E+00      2.9037230390365778
    5 + 2 DET. EVALUATIONS OF       24.10 SEC.
   USER      168.72     DIF      168.72     REF        0.00     SYS        0.64
--------------------------------------------------------------------------------
ewf2p1a6400106cfeci555b33a__usr_bin_f77_r8000_mips4_64_O3_mp_lcomplib.sgimath_41
 -RRLSDE- 1.1, 96/08/02. 1.0, 93/05/18. 1.4, 91/03/13 (VICDF)
 -RRP41C-1.0 98/11/09 RRPSM 4.1.1V1S 96/02/09 -4.1PV1S 4.1PV1 -M_4.1P
 DET:     -7.6262076063431072 10** 37  ENERGY      2.7999999999999998
 DET:      6.5711433071813330 10** 37  ENERGY      2.9999999999999996
 DET:      2.0077284429255240 10** 36  ENERGY      2.9065738535396579
 DET:      3.7022999514429218 10** 33  ENERGY      2.9037282905579587
 DET:     -1.7360127395319718 10** 29  ENERGY      2.9037230387903326
 DET:     -3.8867156868107071 10** 23  ENERGY      2.9037230390365769
 DET:      2.0731391735528364 10** 23  ENERGY      2.9037230390365778
    1  0.2073139173552839E-13  H   0.0000E+00      2.9037230390365778
    5 + 2 DET. EVALUATIONS OF       13.67 SEC.
   USER       95.73     DIF       95.73     REF        0.00     SYS        0.64
--------------------------------------------------------------------------------
ewf2p1a6400106cfeci555b33a__usr_bin_f77_r8000_mips4_64_O3_mp_lcomplib.sgimath_41
 -RRLSDE- 1.1, 96/08/02. 1.0, 93/05/18. 1.4, 91/03/13 (VICDF)
 -RRP41C-1.0 98/11/09 RRPSM 4.1.1V1S 96/02/09 -4.1PV1S 4.1PV1 -M_4.1P
 DET:     -7.6262076063431072 10** 37  ENERGY      2.7999999999999998
 DET:      6.5711433071813330 10** 37  ENERGY      2.9999999999999996
 DET:      2.0077284429255240 10** 36  ENERGY      2.9065738535396579
 DET:      3.7022999514429218 10** 33  ENERGY      2.9037282905579587
 DET:     -1.7360127395319718 10** 29  ENERGY      2.9037230387903326
 DET:     -3.8867156868107071 10** 23  ENERGY      2.9037230390365769
 DET:      2.0731391735528364 10** 23  ENERGY      2.9037230390365778
    1  0.2073139173552839E-13  H   0.0000E+00      2.9037230390365778
    5 + 2 DET. EVALUATIONS OF       10.19 SEC.
   USER       71.35     DIF       71.35     REF        0.00     SYS        0.71
```

```
--------------------------------------------------------------------------------
ewf2p1a6400106cfeci555b33a__usr_bin_f77_r8000_mips4_64_O3_mp_lcomplib.sgimath_41
 -RRLSDE- 1.1, 96/08/02. 1.0, 93/05/18. 1.4, 91/03/13 (VICDF)
 -RRP41C-1.0 98/11/09 RRPSM 4.1.1V1S 96/02/09 -4.1PV1S 4.1PV1 -M_4.1P
 DET:     -7.6262076063431072 10** 37  ENERGY      2.7999999999999998
 DET:      6.5711433071813330 10** 37  ENERGY      2.9999999999999996
 DET:      2.0077284429255240 10** 36  ENERGY      2.9065738535396579
 DET:      3.7022999514429218 10** 33  ENERGY      2.9037282905579587
 DET:     -1.7360127395319718 10** 29  ENERGY      2.9037230387903326
 DET:     -3.8867156868107071 10** 23  ENERGY      2.9037230390365769
 DET:      2.0731391735528364 10** 23  ENERGY      2.9037230390365778
    1  0.2073139173552839E-13  H   0.0000E+00      2.9037230390365778
    5 + 2 DET. EVALUATIONS OF        8.49 SEC.
   USER       59.46     DIF       59.46     REF        0.00     SYS        0.74

--------------------------------------------------------------------------------
ewf2p1a6400106cfeci555b33a__usr_bin_f77_r8000_mips4_64_O3_mp_lcomplib.sgimath_mp
 -RRLSDE- 1.1, 96/08/02. 1.0, 93/05/18. 1.4, 91/03/13 (VICDF)
 -RRP41C-1.0 98/11/09 RRPSM 4.1.1V1S 96/02/09 -4.1PV1S 4.1PV1 -M_4.1P
 DET:     -7.6262076063431072 10** 37  ENERGY      2.7999999999999998
 DET:      6.5711433071813330 10** 37  ENERGY      2.9999999999999996
 DET:      2.0077284429255240 10** 36  ENERGY      2.9065738535396579
 DET:      3.7022999514429218 10** 33  ENERGY      2.9037282905579587
 DET:     -1.7360127395319718 10** 29  ENERGY      2.9037230387903326
 DET:     -3.8867156868107071 10** 23  ENERGY      2.9037230390365769
 DET:      2.0731391735528364 10** 23  ENERGY      2.9037230390365778
    1  0.2073139173552839E-13  H   0.0000E+00      2.9037230390365778
    5 + 2 DET. EVALUATIONS OF       16.92 SEC.
   USER      118.49     DIF      118.49     REF        0.00     SYS        0.89
--------------------------------------------------------------------------------
ewf2p1a6400106cfeci555b33a__usr_bin_f77_r8000_mips4_64_O3_mp_lcomplib.sgimath_mp
 -RRLSDE- 1.1, 96/08/02. 1.0, 93/05/18. 1.4, 91/03/13 (VICDF)
 -RRP41C-1.0 98/11/09 RRPSM 4.1.1V1S 96/02/09 -4.1PV1S 4.1PV1 -M_4.1P
 DET:     -7.6262076063431072 10** 37  ENERGY      2.7999999999999998
 DET:      6.5711433071813330 10** 37  ENERGY      2.9999999999999996
 DET:      2.0077284429255240 10** 36  ENERGY      2.9065738535396579
 DET:      3.7022999514429218 10** 33  ENERGY      2.9037282905579587
 DET:     -1.7360127395319718 10** 29  ENERGY      2.9037230387903326
 DET:     -3.8867156868107071 10** 23  ENERGY      2.9037230390365769
 DET:      2.0731391735528364 10** 23  ENERGY      2.9037230390365778
    1  0.2073139173552839E-13  H   0.0000E+00      2.9037230390365778
    5 + 2 DET. EVALUATIONS OF       10.00 SEC.
   USER       70.04     DIF       70.04     REF        0.00     SYS        0.73
--------------------------------------------------------------------------------
ewf2p1a6400106cfeci555b33a__usr_bin_f77_r8000_mips4_64_O3_mp_lcomplib.sgimath_mp
 -RRLSDE- 1.1, 96/08/02. 1.0, 93/05/18. 1.4, 91/03/13 (VICDF)
 -RRP41C-1.0 98/11/09 RRPSM 4.1.1V1S 96/02/09 -4.1PV1S 4.1PV1 -M_4.1P
 DET:     -7.6262076063431072 10** 37  ENERGY      2.7999999999999998
 DET:      6.5711433071813330 10** 37  ENERGY      2.9999999999999996
 DET:      2.0077284429255240 10** 36  ENERGY      2.9065738535396579
 DET:      3.7022999514429218 10** 33  ENERGY      2.9037282905579587
 DET:     -1.7360127395319718 10** 29  ENERGY      2.9037230387903326
 DET:     -3.8867156868107071 10** 23  ENERGY      2.9037230390365769
 DET:      2.0731391735528364 10** 23  ENERGY      2.9037230390365778
    1  0.2073139173552839E-13  H   0.0000E+00      2.9037230390365778
    5 + 2 DET. EVALUATIONS OF        8.12 SEC.
   USER       56.83     DIF       56.83     REF        0.00     SYS        0.70
```

```
Km = 40, N = 121:

--------------------------------------------------------------------------------
ewf2p1aa40010acfeci555b33a__usr_bin_f77_r8000_mips4_64_O3_lcomplib.sgimath_314.2
 -RRLSDE- 1.1, 96/08/02. 1.0, 93/05/18. 1.4, 91/03/13 (VICDF)
 -RRP31C- 2.0, 96/08/05. -1.0; -RRPSM 3.1.1V1. -3.1V1S -3.0V1S -3.0VS
 DET:     -3.8792406013834837 10**120  ENERGY      2.7999999999999998
 DET:      3.3885222831668278 10**120  ENERGY      2.9999999999999996
 DET:      7.8304525751813507 10**118  ENERGY      2.9058935917048259
 DET:      1.7534098087577488 10**116  ENERGY      2.9037293025731787
 DET:     -7.4871922634738519 10**111  ENERGY      2.9037244482953235
 DET:     -3.9514242780718138 10**106  ENERGY      2.9037244485025955
 DET:      8.1751020746561931 10**105  ENERGY      2.9037244485025968
    1  0.81751020746562O7E-14  H   0.0000E+00      2.9037244485025968
    5 + 2 DET. EVALUATIONS OF      110.93 SEC.
   USER      776.57     DIF      776.57     REF        0.00     SYS        2.37
--------------------------------------------------------------------------------
ewf2p1aa40010acfeci555b33a__usr_bin_f77_r8000_mips4_64_O3_mp_lcomplib.sgimath_mp
 -RRLSDE- 1.1, 96/08/02. 1.0, 93/05/18. 1.4, 91/03/13 (VICDF)
 -RRP41C-1.0 98/11/09 RRPSM 4.1.1V1S 96/02/09 -4.1PV1S 4.1PV1 -M_4.1P
 DET:     -3.8792406013834846 10**120  ENERGY      2.7999999999999998
 DET:      3.3885222831668256 10**120  ENERGY      2.9999999999999996
 DET:      7.8304525751813880 10**118  ENERGY      2.9058935917048259
 DET:      1.7534098086924912 10**116  ENERGY      2.9037293025731787
 DET:     -7.4871715111870980 10**111  ENERGY      2.9037244482953235
 DET:     -7.4444938456006433 10**106  ENERGY      2.9037244485025946
 DET:      8.3176068643449401 10**105  ENERGY      2.9037244485025968
    1  0.8317606864344954E-14  H   0.0000E+00      2.9037244485025968
    5 + 2 DET. EVALUATIONS OF      147.73 SEC.
   USER     1034.16     DIF     1034.16     REF        0.00     SYS       11.68
```

PART III

PARALLELIZATION OF THE CFHHM PROGRAMS ON SPP-1000

October 1995, IJS/NSC SPP-1000 (borg)

The SPP-1000 machine was beset by hardware and software failures. In addition, the best libraries available were not utilized, therefore the results have only qualitative meaning. See Part II for more recent results, which are qualitatively consistent with the present ones.

III.1. Short description of benchmark:

See Appendix III.C for benchmark input file (He atom ground state).
RRPSM is called 61 times. I-loop has MM = 49 iterations.
C3860: vectorization only.
SPP-1000: all routines -O2 -nore -cxpa ... except RRPSM which is as indicated. -nore is used until the reason for program failure without -nore is determined. The program also fails if -cxpa is not included, even if -nore is used.

III.2. Description of initial modifications

To enable parallelization, the I loop (innermost in RRPSM 1.0) was taken out of the loop nest to be the outermost. Physically this means each column of the matrix solution of the system of differential equations was treated independently and in parallel, on each z interval (RRPSM is called once per interval).

Versions 1.1, 4.1 and 4.2 were made on C3860 (see cfhhopt1.txt). Main changes were as follows:

Version	Changes
1.1	Second recursion loop simplified (only QQ appears)
3.0	Streamlined: eliminated calls to BINA and POWA
4.0	Moved I loop outside, RR remains inside, although independent of I
4.1	Factor (p - r - 2) incorporated into a copy af array CNT
4.2	RR outside, factor (p - r - 2) incorporated in RR (RRX)
4.2.1	As 4.2, but RRX 3-dimensional (save half space)
4.2.2	As 4.2.1, but RRX loop parallelized (J outermost)
4.*P*	Explicit parallelization directives
4.*PV*	Explicit parallelization directives, Veclib in innermost

III.3. Results

The programs were transferred from marvin. They mostly fail to run without cxpa, reporting errors like "bus error" or "error unwinding stack." One of the reasons could be that there are DATA statements in many routines. In this test, all of the routines were compiled with -nore (except where noted) although this did not prevent the program from failing occasionally. There is conflicting information in the books whether -nore really prevents the parallelization.

File names were changed a few times. For example, output files of the form ewf2n*_2_*_3_*16a.2ff correspond to old tests without -peel -mrl; the files ewf2n*_2_*_3_*16xa.2ff and ewf2n*_2_*_3x_*16a.2ff should mean the same, namely including -peel -mrl in the rrpsm*.f only.

Table III.1: Execution times. Times are those with cxpa running, as reported by cxpa. The reference value (routine before optimizations) is rrpsm.f (RRPSM 1.0). Compiler options for all routines except RRPSM: fc -O2 -nore -cxpa ...; RRPSM: fc -O3 -nore -cxpa ... (exceptions indicated). (The total CPU time in the output file (routine tempd) is almost twice as large as the cxpa-reported time; cxpa also shows rrpsm to use about 96% CPU time.)

Version	Time						
	C3860 (-O2)			SPP-1000 (-O2, -O3)			
	CPU			No. proc.	CPU		WALL	
	Total	RRPSM	loops		Total	RRPSM	loops	Total	RRPSM	loops
1.0	28	22	22							
1.1	29	22	22							
4.1	49	44	44	1+	81	77		83		
				8	139	134	1*2	221		80*2
4.1P1				32	160	155	145	38		21
				32	158	152	142	37	30	20++ *5
				16	140	134	127	40		27
				16	141	136	129	38	31	24++
				16	142	137	130	34	28	21++*
				16	134	129	122	38	31	24+x
				16	139	134	127	28		14+v
				16	128	124	117	34	28	21+v*
				8	107	103	101	49		42
				8	110	105	103	37	32	29++
4.1P3				8	106	102	100	42		34+r
4.1PV1				32	118	113	103	33		16
				32	112	107	97	33	27	17++
				16	73	68	62	29		17
				16	81	76	69	30	23	17++
				16	79	74	68	30	24	17+x
				16	78	73	67	35	28	22+v
				8	56	51	49	25		17
				8	57	52	50	25	20	18++
4.1P2				32	166	161	152	44		28
				16	137	132	126	44		31
				16	135	130	124	50		37++
				8	110	106	104	46		39
				8	112	107	105	52		45++

4.2	29	24	23	1+	168	164		172		
				8	277	272	234*1	68		39*1
4.2P1				32	442	436	394	76		27
				32	440	434	392	75	68	25++
				16	282	277	248	80		44
				16	280	275	246	64	57	28++
				8	184	180	170	80		65
				8	182	177	167	41	35	25++
4.2.1P1				32	498	492	450	86		36++
				16	303	298	269	78		41++
				8	202	198	187	45		28++
4.2.2P1				32	error unwinding stack					
				16	383	378	281	75		52++ *4
				8	227	223	201	102		89++ *3

*1 Innermost parallel loop (374a). Only loop 335 is reported by fc as parallel, and the innermost loops look like having been simply replicated on all 8 processors (see the cxpa report).
*2 Only loop 284 is parallelized by fc; also, cxpa reports only on its optimization time; the innermost loops are not parallel, and even 284 seems to have been simply replicated. This was the reason to introduce 4.1P1 (directive to force parallelization of 334).
+ Compiled with -O2.
++ Unlimited optimization (fc ... -mrl -peel ...) in rrpsm..., but not in other routines.
++* A repeated "++" run at another time.
+x Unlimited optimization (fc ... -mrl -peel ...) in all routines.
+v Borg Veclib called instead of the CFHH package Veclib sources; unlimited optimization (fc ... -mrl -peel ...) in all other routines.
+v* A repeated "+v" run at another time.
+r -nore removed (reentrant compilation), unlimited optimization. The variable LCALL0 eliminated (assumed that it makes problems with reentrancy, but that may not be true).
*3 The RRX loop is also parallelized and takes 20 (5) sec. The main loop increases enormously against 4.2P1.
*4 The RRX loop takes 92 (12) sec.
*5 On one occasion, first run failed, second (exact repetition) successful.

NOTES

Syspic shows that during parallel execution, the CPU usage is 100% only
on gsm8 (single hypernode); on gsm16, gsm32 it is progressively smaller.
This is reflected in the small speedup on more than 8 processors.

The program sometimes fails which means there is some error
(probably in the machine).
This is independent of whether cxpa is running or not.

Unlimited optimization (-mrl -peel) can help significantly on 8 processors
(see especially 4.2P1 on 8 processors).

On more processors, in all cases, CTI (internode) communication becomes
a severe limiting factor for speed.

4.1P1: total wall time better than with 4.2P1; loops wall time worse.
So 4.1P1 is the better topology of the two; see also 4.2.2P1: because
two loop nests cannot be efficiently parallelized in RRPSM, the gain
in main loop time by 4.2P1 is lost in the auxiliary loop which should not
be parallelized.
On the other hand, 4.1 is bad on C series.

4.2.2P1: two loops parallelized in RRPSM make the time increase; also, a
lot of the time the system is using 100% CPU, which reminds of the situation
when the entire package was naively compiled under -O3.

4.1 and 4.2 could not be compiled under fc -mrl -peel; compiler error.

Appendix III.A

Thread distribution of I from:
ewf2n1a6400106cfeci555b320_2_nr_p_4.1P1_3_nr_p_32b.2ff

(I = 1,... 49; threads 0...32)

-RRPSM -	I	1	THREAD	0
-RRPSM -	I	2	THREAD	1
-RRPSM -	I	3	THREAD	2
-RRPSM -	I	4	THREAD	3
-RRPSM -	I	5	THREAD	4
-RRPSM -	I	6	THREAD	5
-RRPSM -	I	7	THREAD	6
-RRPSM -	I	8	THREAD	7
-RRPSM -	I	9	THREAD	8
-RRPSM -	I	10	THREAD	9
-RRPSM -	I	11	THREAD	10
-RRPSM -	I	12	THREAD	11
-RRPSM -	I	13	THREAD	12
-RRPSM -	I	14	THREAD	13
-RRPSM -	I	15	THREAD	14
-RRPSM -	I	16	THREAD	15
-RRPSM -	I	17	THREAD	15
-RRPSM -	I	18	THREAD	16
-RRPSM -	I	19	THREAD	16
-RRPSM -	I	20	THREAD	17
-RRPSM -	I	21	THREAD	17
-RRPSM -	I	22	THREAD	18
-RRPSM -	I	23	THREAD	18
-RRPSM -	I	24	THREAD	19
-RRPSM -	I	25	THREAD	19
-RRPSM -	I	26	THREAD	20
-RRPSM -	I	27	THREAD	20
-RRPSM -	I	28	THREAD	21
-RRPSM -	I	29	THREAD	21
-RRPSM -	I	30	THREAD	22
-RRPSM -	I	31	THREAD	22
-RRPSM -	I	32	THREAD	23
-RRPSM -	I	33	THREAD	23
-RRPSM -	I	34	THREAD	24
-RRPSM -	I	35	THREAD	24
-RRPSM -	I	36	THREAD	25
-RRPSM -	I	37	THREAD	25
-RRPSM -	I	38	THREAD	26
-RRPSM -	I	39	THREAD	26
-RRPSM -	I	40	THREAD	27
-RRPSM -	I	41	THREAD	27
-RRPSM -	I	42	THREAD	28
-RRPSM -	I	43	THREAD	28
-RRPSM -	I	44	THREAD	29
-RRPSM -	I	45	THREAD	29
-RRPSM -	I	46	THREAD	30
-RRPSM -	I	47	THREAD	30
-RRPSM -	I	48	THREAD	31
-RRPSM -	I	49	THREAD	31

Appendix III.B

File: /users/krivec/cfhh/pd/opt_rrpsm_borg.csh:

```
# Borg rrpsm optimization ‘‘make’’ routine.
# opt_rrpsm: exchanges of rrpsm*.o in ol_O2....a, relink.
# rrpsm is compiled with fc -$opt1 -$nore1 -$pax1,
# other routines with    fc -$opt  -$nore  -$pax.
# (nore should be "nr" or empty, pax should be "p" or empty.)
# CPU and wall times may vary due to different loads and profiling.
# R. Krivec 95/10/17.
# R. Krivec 95/10/20.
# R. Krivec 95/10/27.

# set verbose
  unset rrpsmonly
  unset linkonly
  unset setonly

  set rrpsmonly  # careful: the first time ddaa_2.0 has to be compiled
# set linkonly
# set setonly

  set vl=vb
  set eex0=24p0s

  set opt=2x
  set nore=nr
  set pax=p
  set lib="ol_‘echo ${opt} ${nore} ${pax} | sed ’s/ /_/g’‘"

  set ver=4.1P1
  set opt1=3x
  set nore1=nr
  set pax1=p
  set eex="_opt_rrpsm_‘echo $lib $ver $opt1 $nore1 $pax1 $vl | sed ’s/ /_/g’‘"

  echo "--" $vl
  echo "--" $eex0 $opt $nore $pax
  echo "--" $ver $opt1 $nore1 $pax1
  echo -n "Return to continue... "; set zzz=$<
  echo "--" $lib
  echo "--" $eex
  echo -n "Return to continue... "; set zzz=$<
  if ( $vl == ’vb’ ) then
      set lib=${lib}_nv           # modify lib ######################
      if ( ! -e ${lib}.a ) exit
      echo "--" $lib
      echo -n "Return to continue... "; set zzz=$<
  endif
  if ( $test == ’test’ ) goto _TEST


  if ( $?setonly ) then
      unset setonly
      exit
  endif

  if ( $?linkonly ) goto _LINK

# Cleanup the libraries
  ar tv ${lib}.a | grep -E ’ddaa|rrpsm\.|rrpsm_’
  ar dv ${lib}.a rrpsm.o |& grep -v ’not found’
  ar dv ${lib}.a rrpsm_*.o |& grep -v ’not found’
  ar dv ${lib}.a ‘echo rrpsm_*.o | sed ’s/\.o/\./g’‘ |& grep -v ’not found’
                          # the above works for one character too much
  if ( $vl == ’vb’ ) then
      source ewf2n_veclib_lib_remove.csh
```

```
  endif
  if ( $?rrpsmonly ) goto _C1
      ar dv ${lib}.a ddaa.o |& grep -v 'not found'
      ar dv ${lib}.a ddaa*.o |& grep -v 'not found'
  _C1:
  ar tv ${lib}.a | grep -E 'ddaa|rrpsm\.|rrpsm_'

  pro rrpsm_${ver}
  echo $pro
  alias zzz fc${opt1}${nore1}${pax1}
  zzz
  ~/bin/para.csh $pro
  xpr

  if ( $?rrpsmonly ) goto _LINK

  _TEST:
  if ( $ver == '1.0' || $ver == '1.1' ) then
      pro ddaa
  else
      pro ddaa_2.0
  endif
  echo $pro
  alias zzz fc${opt}${nore}${pax}
  zzz
  xpr

  pro ewf2n
  echo $pro
  alias zzz fc${opt}${nore}${pax}
  zzz

  _LINK:
  ar tv ${lib}.a | grep -E 'ddaa|rrpsm\.|rrpsm_'
  pro ewf2n
  alias zzz ldu${vl}${pax}
  zzz

# grep ', rrpsm' ewf2n.m$eex
  kk ewf2n.e$eex ewf2n.m$eex
  unset rrpsmonly
  unset linkonly
```

Appendix III.C

File: /users/krivec/cfhh/inp/ewf2n1a_opt_rrpsm.i:

```
95/10/06 He opt_rrpsm
FFFFFFFF        LTDD,LTDDR,LTVICD,LTRRPS,LPRBIN,LPRZ0M,LTCPU,LTCRU
TFFTFT          LPRCV0, LPRXX, LPRCV, *LPRCVM, LPRMAT, LPRWF
TFT             LTZE, LTZEIF, LXLAST (IPROG = 200)
T               LTDET
FF              LDOUBL, LDOUBM
9999            IINTPR
  20 200        LM, IPROG
   0            IENXL (LMOD = 5, 6 ONLY)
   0            I0D
   0  49   1                            IPWUU, NSU, INUSEL
  20  16                                NM, JMAX
    0.50    0.50  9999.0                BC0, BC1, RHOUW
2.9000          3.0000                  A, B
  +1.0000000D-14  0.10000000D+00  5.00000000D+00    EPSP, TOL, DZU
    0.50    5.00  120.00                ZM, ZMATCH, ZTOP
   0                                    NHH
   2                                    LMOD
2.9                         3.00            1.00    ENXL, ENXU, ENXS
   1                                    IPIO (0: M1M-, 1: M2M-)
    0.00  999.00   1   1                Z0MTTT, Z0MTTD, IPRTTT, IPCTTT
-------------------------------------------------------------------------------
```

File: /users/krivec/cfhh/inp/ewf2n1a_opt_rrpsm_borg.c:

```
#!/bin/csh -f

  echo $ver                    # this must be set
  echo $opt $nore $pax         # this must be set
  echo $opt1 $nore1 $pax1      # this must be set
  echo $vl                     # this must be set
  echo $paxe                   # this must be set
  echo $repv                   # this must be set
  echo -n "Return to continue... "; set zzz=$<

  set xxx="‘echo $opt $nore $pax | sed ’s/ /_/g’‘"
  set xxx1="‘echo $ver $opt1 $nore1 $pax1 $vl | sed ’s/ /_/g’‘"


  set pi1 = ’xxx’; set fo3 = ’xxx’; set fo4 = ’xxx’
  set pi2 = ’xxx’; set fo5 = ’xxx’; set fo6 = ’xxx’
  set rr = ’xxx’; set best = ’xxx’; set work = ’xxx’


  set rr = 0    # 0: original run;
                # 1: restart (LMOD = 5, --> w.f.);
                # 2: re-restart (LMOD = 51, -->  w.f.);
                # 3: tabulation (LMOD = 7, E from iteration file).
  set xer = 0   # 0: previous run wrote best E value file
                # 1: previous run stopped before writing best E value file
                # (this overrides value of best if previous run had error)


  if ($rr == 1 || $rr == 2) set best = 1  # restart E: best value
  if ($rr == 3) set best = 0              # tabulation E; iteration file
  if ($xer == 1) set best = 0             # override best: use iteration file


  set work = 1  # 1: ewf2-
                # 2: av3m-
                # 3: both


  set fi  = ’1a_opt_rrpsm’                #  8
  set fo1 = ’1a64001’                     #  9
  set fo2 = "06cfeci555b320_${xxx}_${xxx1}_${repv}"     # 10
  echo    $fo2
  echo -n "Return to continue... "; set zzz = $<

  if ($rr >= 1) then
    set pi1 = ’ewf2n’                     # 11   initial run
    set fo3 = ’1aa4001’                   # 12
    set fo4 = ’0acfec5555b33a’            # 13
    echo ’---- pi1: ’ $pi1
  endif

  if ($rr == 2) then
    set pi2 = ’ewf2n’                     # 14   previous restart run
    set fo5 = ’1aa4001’                   # 15
    set fo6 = ’0acfec5555b33a’            # 16
    echo ’---- pi2: ’ $pi2
  endif

  set de  = "~/cfhh/pd"                   #  1
  set di  = "~/cfhh/inp"                  #  2
  set do  = "~/cfhh/out/opt_rrpsm"
  set ds  = "~/cfhh/out/opt_rrpsm"
  set pe  = ’ewf2n’                       #  5
  set pi  = ’m2mk’                        #  6
  set fe  = "e_opt_rrpsm_ol_${xxx}_${xxx1}" #  7
```

```
if ($work == 1 || $work == 3) then
    echo '---- work' $work

    if ( $paxe != 'p' ) set paxe = '0'
    $di/${pe}00${paxe}.csh \
        $de $di $do $ds $pe $pi $fe $fi $fo1 $fo2 $pi1 $fo3 $fo4\
        $pi2 $fo5 $fo6 $rr $best

    #   1   2   3   4   5   6   7   8   9    10   11   12   13
    #   14   15   16   17  18

endif
set zzzrep=$do/ewf2n${fo1}${fo2}.report
if ( -e ewf2n.report ) mv ewf2n.report $zzzrep

if ($work > 1) then

    set do1 = ''
    set pi = $pe
    set pi1 = 'n'  # see $pi
    set pe = 'av3mj'
    set fe = 'e'

    set fi1 = '2222224'
    set fi2 = 'a'
    set fi  = 'md'$fi1$fi2
    set fo3  = '0a'$fi1${fi2}$pi1

    $di/${pe}000.c $de $di $do $ds $pe $pi $fe $fi $fo1 $fo2 $fo3 $do1

    #               1   2   3   4   5   6   7   8    9   10   11   12

endif
```

Appendix III.D

Optimized versions of RRPSM with comments filtered out.

File: rrpsm_1.0.fnoc

```
      SUBROUTINE    RRPSM(NCDS, NCDP, NCDCN, NDBIN1, NDBIN2, NDZ0MP,
     1                    WS, WD, MM, MU, IPXU, IPOFF, Z0M, JMAX, AA,
     2                    RHOUW, PP, QQ, RR, BIN, Z0MPOW,
     3                    LTEST, LPRBIN, LPRZ0M, NOUT,
     4                    CNT)
      IMPLICIT      REAL*8 (A - H, O - Z)
      LOGICAL       LTEST, LPRBIN, LPRZ0M
      DIMENSION     WS(NCDS,NCDS,NCDP), WD(NCDS,NCDS,NCDP),
     1              MU(NCDS),
     2              PP(NCDS,NCDS,NCDCN), QQ(NCDS,NCDS,NCDCN),
     3              RR(NCDS,NCDS), BIN(NDBIN1,NDBIN2), Z0MPOW(NDZ0MP),
     4              CNT(NCDS,NCDS,NCDCN)
      CHARACTER*6   NAME / 'RRPSM ' /
      CHARACTER*60  VERS
     */ ' 1.0, 93/05/14 (RRPSL 2.0) STREAMLINED; DISTR. M1 LOOP      ' /
      LOGICAL       LCALL0
      DATA          LCALL0 / .TRUE. /
      DATA          ZERO, ONE, TWO / 0.D0, 1.D0, 2.D0 /
      IF (LCALL0) WRITE (NOUT,4000) NAME, VERS
      IF (MM      .GT. NCDS)       CALL ERR2(NAME, 1, 1, NOUT)
      IF (IPXU    .GT. NCDP)       CALL ERR2(NAME, 2, 1, NOUT)
      IF (JMAX+1  .GT. NCDCN)      CALL ERR2(NAME, 3, 1, NOUT)
      IF (ABS(AA) .LT. CDXTOL())   CALL ERR2(NAME, 4, 1, NOUT)
      IF (LCALL0 .AND. IPOFF .GT. 2) CALL ERR2(NAME, 11, 0, NOUT)
      IF (LCALL0 .AND. JMAX .LT. 2) CALL ERR2(NAME, 21, 0, NOUT)
      IF (LCALL0) LCALL0 = .FALSE.
      IF (JMAX .LT. 2) RETURN
      IF (RHOUW .GE. ZERO) ZUW = TWO * AA * RHOUW
      IF (RHOUW .LT. ZERO) ZUW = -RHOUW
      JMAX1 = JMAX + 1
      ZZZ4  = ONE / (TWO * AA)
      IPU = IPXU - IPOFF
      JRU = JMAX - 2 + 1
      CALL    BINA(NDBIN1, NDBIN2, IPU, JRU, BIN, LPRBIN, NOUT)
      IPOWL = - JMAX
      IPOWU = IPU
      CALL    POWA(NDZ0MP, IPOWL, IPOWU, Z0M, LPRZ0M, NOUT,
     1             Z0MPOW, NZ0MOF)
      DO  68  M1 = 1,JMAX
          JR = M1 - 2
          ZZZ51 = BIN(-1+3,JR+1+1) * Z0MPOW(-JR-2+NZ0MOF)
          DO  22  I = 1,MM
              DO  20  J = 1,MM
                  PP(J,I,M1) = ZERO
   20         CONTINUE
   22     CONTINUE
          DO  24  I = 1,MM
              PP(I,I,M1) = ZZZ51
   24     CONTINUE
          IF (Z0M .LE. ZUW) THEN
              IPXL = JR + 1 + IPOFF
              IF (IPXL .LE. IPXU) THEN
                  DO  34  IPX = IPXL,IPXU
                      IP = IPX - IPOFF
                      ZZZ52 = - BIN(IP+3,JR+1+1)
     1                        * Z0MPOW(IP-JR-1+NZ0MOF)
     2                        * ZZZ4 ** (IP+1)
                      DO  32  I = 1,MM
                          DO  30  J = 1,MM
                              PP(J,I,M1) = PP(J,I,M1)
     1                                     + ZZZ52 * WD(J,I,IPX)
   30                     CONTINUE
```

```
   32               CONTINUE
   34           CONTINUE
            ENDIF
        ENDIF
        DO  42  I = 1,MM
            DO  40  J = 1,MM
                QQ(J,I,M1) = ZERO
   40       CONTINUE
   42   CONTINUE
        IF (JR .LT. 0) GO TO  66
            IF (JR .EQ. 0) THEN
                ZZZ61 = -0.25D0
                DO  50  I = 1,MM
                    QQ(I,I,M1) = ZZZ61
   50           CONTINUE
            ENDIF
            IF (JR .GE. 0) THEN
                ZZZ62 = -BIN(-2+3,JR+1) * Z0MPOW(-JR-2+NZ0MOF)
                DO  52  I = 1,MM
                    QQ(I,I,M1) = QQ(I,I,M1) + ZZZ62 * MU(I) ** 2
   52           CONTINUE
            ENDIF
        IF (Z0M .GT. ZUW) GO TO  66
            IPXL = -1 + IPOFF
            IF (IPXL .GT. IPXU) CALL ERR2(NAME, 31, 1, NOUT)
            DO  64  IPX = IPXL,IPXU
                IP = IPX - IPOFF
            IF (IP .GE. 0  .AND.  JR .GT. IP) GO TO  64
                ZZZ63 = - BIN(IP+3,JR+1) * Z0MPOW(IP-JR+NZ0MOF)
     1                  * ZZZ4 ** (IP+2)
                DO  62  I = 1,MM
                    DO  60  J = 1,MM
                        QQ(J,I,M1) = QQ(J,I,M1) + ZZZ63 * WS(J,I,IPX)
   60               CONTINUE
   62           CONTINUE
   64       CONTINUE
   66   CONTINUE
   68 CONTINUE
      DO  124  N1 = 3,JMAX1
        DO  72  J = 1,MM
            DO  70  I = 1,MM
                CNT(I,J,N1) = ZERO
   70       CONTINUE
   72   CONTINUE
        JP = N1 - 1
        JP1 = JP - 1
        DO  90  M1 = 1,JP1
            JR    = M1 - 2
            ZZZ72 = JP - JR - 2
            DO  82  K = 1,MM
                DO  80  J = 1,MM
                    RR(J,K) = ZZZ72 * PP(J,K,M1) + QQ(J,K,M1)
   80           CONTINUE
   82       CONTINUE
            DO  88  J = 1,MM
                DO  86  I = 1,MM
                    DO  84  K = 1,MM
                        CNT(I,J,N1) = CNT(I,J,N1)
     1                      + CNT(I,K,N1-JR-2) * RR(J,K)
   84               CONTINUE
   86           CONTINUE
   88       CONTINUE
   90   CONTINUE
        M1    = JP
        JR    = M1 - 2
        ZZZ72 = JP - JR - 2
        DO  104  K = 1,MM
            DO  102  J = 1,MM
                RR(J,K) = ZZZ72 * PP(J,K,M1) + QQ(J,K,M1)
  102       CONTINUE
  104   CONTINUE
        DO  108  J = 1,MM
```

```
              DO  106  I = 1,MM
                  CNT(I,J,N1) = CNT(I,J,N1) + RR(J,I)
 106          CONTINUE
 108      CONTINUE
          ZZZ71 = - ONE / DFLOAT(JP * (JP - 1))
          DO  122  J = 1,MM
              DO  120  I = 1,MM
                  CNT(I,J,N1) = ZZZ71 * CNT(I,J,N1)
 120          CONTINUE
 122      CONTINUE
 124  CONTINUE
      IF (.NOT.LTEST) RETURN
      WRITE (NOUT,4100) NAME, Z0M
      DO  132  K = 1,JMAX1
          WRITE (NOUT,4110) K
          DO  130  I = 1,MM
              WRITE (NOUT,4120) (CNT(I,J,K), J = 1,MM)
 130      CONTINUE
 132  CONTINUE
      RETURN
4000  FORMAT (
     A  2H -, A6, 1H-, A60)
4100  FORMAT (//, 1H , 80(1H-), /,
     A  1H , 1H-, 2A4, 1H-, 4HZ0M , F12.6)
4110  FORMAT (/,
     A  1H , 4H  K , I4)
4120  FORMAT (
     A (1H , 8X, 4F16.8))
      END
```

File: rrpsm_4.0.fnoc

```
      SUBROUTINE    RRPSM(NCDS, NCDP, NCDCN, NDBIN1, NDBIN2,
     1                    WS, WD, MM, MU, IPXU, IPOFF, Z0M, JMAX, AA,
     2                    RHOUW, PP, QQ, RR, BIN,
     3                    LTEST, NOUT, CNT)
      IMPLICIT      REAL*8 (A - H, O - Z)
      LOGICAL       LTEST
      DIMENSION     WS(NCDS,NCDS,NCDP), WD(NCDS,NCDS,NCDP),
     1              MU(NCDS),
     2              PP(NCDS,NCDS,NCDCN), QQ(NCDS,NCDS,NCDCN),
     3              RR(NCDS,NCDS), BIN(NDBIN1,NDBIN2),
     4              CNT(NCDS,NCDS,NCDCN)
      CHARACTER*6   NAME / 'RRPSM ' /
      CHARACTER*60  VERS
     */ ' 4.0, 95/10/08 -M_3.0                                          ' /
      PARAMETER     (ZERO = 0.D0, ONE = 1.D0, TWO = 2.D0)
      PARAMETER     (NPDS = 49, NPDCN = 22)
      DIMENSION     CNTX(NPDS,NPDCN)
      LOGICAL       LCALL0
      DATA          LCALL0 / .TRUE. /
      IF (LCALL0) WRITE (NOUT,4000) NAME, VERS
      IF (MM      .GT. NCDS)       CALL ERR2(NAME, 1, 1, NOUT)
      IF (IPXU    .GT. NCDP)       CALL ERR2(NAME, 2, 1, NOUT)
      IF (JMAX+1  .GT. NCDCN)      CALL ERR2(NAME, 3, 1, NOUT)
      IF (ABS(AA) .LT. CDXTOL())   CALL ERR2(NAME, 4, 1, NOUT)
      IF (NPDS    .NE. NCDS)       CALL ERR2(NAME, 5, 1, NOUT)
      IF (NPDCN   .NE. NCDCN)      CALL ERR2(NAME, 6, 1, NOUT)
      IF (LCALL0 .AND. IPOFF .GT. 2) CALL ERR2(NAME, 11, 0, NOUT)
      IF (LCALL0 .AND. JMAX .LT. 2) CALL ERR2(NAME, 21, 0, NOUT)
      IF (LCALL0) LCALL0 = .FALSE.
      IF (JMAX .LT. 2) RETURN
      IF (RHOUW .GE. ZERO) ZUW = TWO * AA * RHOUW
      IF (RHOUW .LT. ZERO) ZUW = -RHOUW
      JMAX1 = JMAX + 1
      RHO0  = Z0M / (TWO * AA)
      DO  68  M1 = 1,JMAX
          JR     = M1 - 2
          Z0MIR2 = ONE / (Z0M ** (JR + 2))
          ZZZA = BIN(-1+3,JR+1+1) * Z0MIR2
          DO  22  I = 1,MM
              DO  20  J = 1,MM
                  PP(J,I,M1) = ZERO
20            CONTINUE
22        CONTINUE
          DO  24  I = 1,MM
              PP(I,I,M1) = ZZZA
24        CONTINUE
          IF (Z0M .LE. ZUW) THEN
              IPXL = JR + 1 + IPOFF
              IF (IPXL .LE. IPXU) THEN
                  ZZZB0 = Z0MIR2 * RHO0 ** (IPXL - IPOFF + 1)
                  DO  34  IPX = IPXL,IPXU
                      IP = IPX - IPOFF
                      ZZZB = - BIN(IP+3,JR+1+1) * ZZZB0
                      DO  32  I = 1,MM
                          DO  30  J = 1,MM
                              PP(J,I,M1) = PP(J,I,M1)
     1                                     + ZZZB * WD(J,I,IPX)
30                        CONTINUE
32                    CONTINUE
                      ZZZB0 = ZZZB0 * RHO0
34                CONTINUE
              ENDIF
          ENDIF
          DO  42  I = 1,MM
              DO  40  J = 1,MM
                  QQ(J,I,M1) = ZERO
40            CONTINUE
42        CONTINUE
```

```
          IF (JR .LT. 0) GO TO  66
              IF (JR .EQ. 0) THEN
                  ZZZC = -0.25D0
                  DO  50  I = 1,MM
                      QQ(I,I,M1) = ZZZC
   50             CONTINUE
              ENDIF
              IF (JR .GE. 0) THEN
                  ZZZD = -BIN(-2+3,JR+1) * Z0MIR2
                  DO  52  I = 1,MM
                      QQ(I,I,M1) = QQ(I,I,M1) + ZZZD * MU(I) ** 2
   52             CONTINUE
              ENDIF
          IF (Z0M .GT. ZUW) GO TO  66
              IPXL = -1 + IPOFF
              IF (IPXL .GT. IPXU) CALL ERR2(NAME, 31, 1, NOUT)
              ZZZE0 = Z0MIR2 * RHO0 ** (IPXL - IPOFF + 2)
              DO  65  IPX = IPXL,IPXU
                  IP = IPX - IPOFF
                  IF (IP .GE. 0  .AND.  JR .GT. IP) GO TO  64
                      ZZZE = - BIN(IP+3,JR+1) * ZZZE0
                      DO  62  I = 1,MM
                          DO  60  J = 1,MM
                              QQ(J,I,M1) = QQ(J,I,M1)
     &                                     + ZZZE * WS(J,I,IPX)
   60                     CONTINUE
   62                 CONTINUE
   64             CONTINUE
                  ZZZE0 = ZZZE0 * RHO0
   65         CONTINUE
   66     CONTINUE
   68 CONTINUE
      DO  122  I = 1,MM
          DO  70  J = 1,MM
              CNTX(J,2) = CNT(I,J,2)
   70     CONTINUE
          DO  120  N1 = 3,JMAX1
              JP = N1 - 1
              DO  72  J = 1,MM
                  CNTX(J,N1) = ZERO
   72         CONTINUE
              JP1 = JP - 1
              DO  90  M1 = 1,JP1
                  JR    = M1 - 2
                  ZZZ72 = JP - JR - 2
                  DO  82  K = 1,MM
                      DO  80  J = 1,MM
                          RR(J,K) = ZZZ72 * PP(J,K,M1) + QQ(J,K,M1)
   80                 CONTINUE
   82             CONTINUE
                  DO  86  K = 1,MM
                      DO  84  J = 1,MM
                          CNTX(J,N1) = CNTX(J,N1)
     1                        + RR(J,K) * CNTX(K,N1-JR-2)
   84                 CONTINUE
   86             CONTINUE
   90         CONTINUE
              M1 = JP
              DO  92  J = 1,MM
                  CNTX(J,N1) = CNTX(J,N1) + QQ(J,I,M1)
   92         CONTINUE
              ZZZ71 = - ONE / DFLOAT(JP * (JP - 1))
              DO  100  J = 1,MM
                  CNTX(J,N1) = ZZZ71 * CNTX(J,N1)
  100         CONTINUE
              DO  110  J = 1,MM
                  CNT(I,J,N1) = CNTX(J,N1)
  110         CONTINUE
  120     CONTINUE
  122 CONTINUE
      IF (.NOT.LTEST) RETURN
      WRITE (NOUT,4100) NAME, Z0M
```

```
      DO  132  K = 1,JMAX1
          WRITE (NOUT,4110) K
          DO  130  I = 1,MM
              WRITE (NOUT,4120) (CNT(I,J,K), J = 1,MM)
 130      CONTINUE
 132  CONTINUE
      RETURN
4000  FORMAT (
     A  2H -, A6, 1H-, A60)
4100  FORMAT (//, 1H , 80(1H-), /,
     A  1H , 1H-, 2A4, 1H-, 4HZ0M , F12.6)
4110  FORMAT (/,
     A  1H , 4H  K , I4)
4120  FORMAT (
     A (1H , 8X, 4F16.8))
      END
```

File: rrpsm_4.1P1.fnoc

```
      SUBROUTINE    RRPSM(NCDS, NCDP, NCDCN, NDBIN1, NDBIN2,
     1                    WS, WD, MM, MU, IPXU, IPOFF, Z0M, JMAX, AA,
     2                    RHOUW, PP, QQ, RR, BIN,
     3                    LTEST, NOUT, CNT)
      IMPLICIT      REAL*8 (A - H, O - Z)
      LOGICAL       LTEST
      DIMENSION     WS(NCDS,NCDS,NCDP), WD(NCDS,NCDS,NCDP),
     1              MU(NCDS),
     2              PP(NCDS,NCDS,NCDCN), QQ(NCDS,NCDS,NCDCN),
     3              RR(NCDS,NCDS), BIN(NDBIN1,NDBIN2),
     4              CNT(NCDS,NCDS,NCDCN)
      CHARACTER*6   NAME / 'RRPSM ' /
      CHARACTER*60  VERS
     */ ' 4.1P1, 95/10/19 -M_4.1 (SEE _4.2P1).                        ' /
      PARAMETER     (ZERO = 0.D0, ONE = 1.D0, TWO = 2.D0)
      PARAMETER     (NPDS = 49, NPDCN = 22)
      DIMENSION     CNTX(NPDS,NPDCN), CNTY(NPDS,NPDCN)
      LOGICAL       LCALL0
      DATA          LCALL0 / .TRUE. /
      IF (LCALL0) WRITE (NOUT,4000) NAME, VERS
      IF (MM      .GT. NCDS)       CALL ERR2(NAME, 1, 1, NOUT)
      IF (IPXU    .GT. NCDP)       CALL ERR2(NAME, 2, 1, NOUT)
      IF (JMAX+1  .GT. NCDCN)      CALL ERR2(NAME, 3, 1, NOUT)
      IF (ABS(AA) .LT. CDXTOL())   CALL ERR2(NAME, 4, 1, NOUT)
      IF (NPDS    .NE. NCDS)       CALL ERR2(NAME, 5, 1, NOUT)
      IF (NPDCN   .NE. NCDCN)      CALL ERR2(NAME, 6, 1, NOUT)
      IF (LCALL0 .AND. IPOFF .GT. 2) CALL ERR2(NAME, 11, 0, NOUT)
      IF (LCALL0 .AND. JMAX .LT. 2) CALL ERR2(NAME, 21, 0, NOUT)
      IF (LCALL0) LCALL0 = .FALSE.
      IF (JMAX .LT. 2) RETURN
      IF (RHOUW .GE. ZERO) ZUW = TWO * AA * RHOUW
      IF (RHOUW .LT. ZERO) ZUW = -RHOUW
      JMAX1 = JMAX + 1
      RHO0  = Z0M / (TWO * AA)
C$DIR NO_PARALLEL
      DO  68  M1 = 1,JMAX
          JR     = M1 - 2
          Z0MIR2 = ONE / (Z0M ** (JR + 2))
          ZZZA = BIN(-1+3,JR+1+1) * Z0MIR2
C$DIR NO_PARALLEL
          DO  22  I = 1,MM
C$DIR NO_PARALLEL
              DO  20  J = 1,MM
                  PP(J,I,M1) = ZERO
   20         CONTINUE
   22     CONTINUE
C$DIR NO_PARALLEL
          DO  24  I = 1,MM
              PP(I,I,M1) = ZZZA
   24     CONTINUE
          IF (Z0M .LE. ZUW) THEN
              IPXL = JR + 1 + IPOFF
              IF (IPXL .LE. IPXU) THEN
                  ZZZB0 = Z0MIR2 * RHO0 ** (IPXL - IPOFF + 1)
C$DIR NO_PARALLEL
                  DO  34  IPX = IPXL,IPXU
                      IP = IPX - IPOFF
                      ZZZB = - BIN(IP+3,JR+1+1) * ZZZB0
C$DIR NO_PARALLEL
                      DO  32  I = 1,MM
C$DIR NO_PARALLEL
                          DO  30  J = 1,MM
                              PP(J,I,M1) = PP(J,I,M1)
     1                                   + ZZZB * WD(J,I,IPX)
   30                     CONTINUE
   32                 CONTINUE
                      ZZZB0 = ZZZB0 * RHO0
   34             CONTINUE
```

```
              ENDIF
            ENDIF
C$DIR NO_PARALLEL
            DO  42  I = 1,MM
C$DIR NO_PARALLEL
                DO  40  J = 1,MM
                    QQ(J,I,M1) = ZERO
   40           CONTINUE
   42       CONTINUE
            IF (JR .LT. 0) GO TO  66
                IF (JR .EQ. 0) THEN
                    ZZZC = -0.25D0
C$DIR NO_PARALLEL
                    DO  50  I = 1,MM
                        QQ(I,I,M1) = ZZZC
   50               CONTINUE
                ENDIF
                IF (JR .GE. 0) THEN
                    ZZZD = -BIN(-2+3,JR+1) * Z0MIR2
C$DIR NO_PARALLEL
                    DO  52  I = 1,MM
                        QQ(I,I,M1) = QQ(I,I,M1) + ZZZD * MU(I) ** 2
   52               CONTINUE
                ENDIF
            IF (Z0M .GT. ZUW) GO TO  66
                IPXL = -1 + IPOFF
                IF (IPXL .GT. IPXU) CALL ERR2(NAME, 31, 1, NOUT)
                ZZZE0 = Z0MIR2 * RHO0 ** (IPXL - IPOFF + 2)
C$DIR NO_PARALLEL
                DO  65  IPX = IPXL,IPXU
                    IP = IPX - IPOFF
                    IF (IP .GE. 0  .AND.  JR .GT. IP) GO TO  64
                        ZZZE = - BIN(IP+3,JR+1) * ZZZE0
C$DIR NO_PARALLEL
                        DO  62  I = 1,MM
C$DIR NO_PARALLEL
                            DO  60  J = 1,MM
                                QQ(J,I,M1) = QQ(J,I,M1)
     &                                       + ZZZE * WS(J,I,IPX)
   60                       CONTINUE
   62                   CONTINUE
   64               CONTINUE
                    ZZZE0 = ZZZE0 * RHO0
   65           CONTINUE
   66       CONTINUE
   68 CONTINUE
C$DIR LOOP_PARALLEL
C$DIR LOOP_PRIVATE(CNTX,CNTY,J,N1,JP,JP1,M1,JR,K,ZZZ71)
      DO  122  I = 1,MM
C$DIR NO_PARALLEL
            DO  70  J = 1,MM
                CNTX(J,2) = CNT(I,J,2)
   70       CONTINUE
C$DIR NO_PARALLEL
            DO  71  J = 1,MM
                CNTY(J,2) = CNTX(J,2)
   71       CONTINUE
C$DIR NO_PARALLEL
            DO  120  N1 = 3,JMAX1
                JP = N1 - 1
C$DIR NO_PARALLEL
                DO  72  J = 1,MM
                    CNTX(J,N1) = ZERO
   72           CONTINUE
                JP1 = JP - 1
C$DIR NO_PARALLEL
                DO  90  M1 = 1,JP1
                    JR = M1 - 2
C$DIR NO_PARALLEL
                    DO  82  K = 1,MM
C$DIR NO_PARALLEL
                        DO  80  J = 1,MM
```

```
                            CNTX(J,N1) = CNTX(J,N1)
     1                          + PP(J,K,M1) * CNTY(K,N1-JR-2)
   80                    CONTINUE
   82                CONTINUE
C$DIR NO_PARALLEL
                     DO  86  K = 1,MM
C$DIR NO_PARALLEL
                         DO  84  J = 1,MM
                             CNTX(J,N1) = CNTX(J,N1)
     1                           + QQ(J,K,M1) * CNTX(K,N1-JR-2)
   84                    CONTINUE
   86                CONTINUE
   90            CONTINUE
                 M1 = JP
C$DIR NO_PARALLEL
                 DO  92  J = 1,MM
                     CNTX(J,N1) = CNTX(J,N1) + QQ(J,I,M1)
   92            CONTINUE
                 ZZZ71 = - ONE / DFLOAT(JP * (JP - 1))
C$DIR NO_PARALLEL
                 DO  100  J = 1,MM
                     CNTX(J,N1) = ZZZ71 * CNTX(J,N1)
  100            CONTINUE
C$DIR NO_PARALLEL
                 DO  102  J = 1,MM
                     CNTY(J,N1) = (N1 - 1) * CNTX(J,N1)
  102            CONTINUE
C$DIR NO_PARALLEL
                 DO  110  J = 1,MM
                     CNT(I,J,N1) = CNTX(J,N1)
  110            CONTINUE
  120        CONTINUE
  122 CONTINUE
      IF (.NOT.LTEST) RETURN
      WRITE (NOUT,4100) NAME, Z0M
C$DIR NO_PARALLEL
      DO  132  K = 1,JMAX1
          WRITE (NOUT,4110) K
C$DIR NO_PARALLEL
          DO  130  I = 1,MM
              WRITE (NOUT,4120) (CNT(I,J,K), J = 1,MM)
  130     CONTINUE
  132 CONTINUE
      RETURN
 4000 FORMAT (
     A  2H -, A6, 1H-, A60)
 4100 FORMAT (//, 1H , 80(1H-), /,
     A  1H , 1H-,  A6, 1H-, 4HZ0M , F12.6)
 4110 FORMAT (/,
     A  1H , 4H  K , I4)
 4120 FORMAT (
     A (1H , 8X, 4F16.8))
 6000 FORMAT (
     A  1H , 1H-,  A6, 1H-, 4H   I, I8, 8H  THREAD, I8)
      END
```

File: rrpsm_4.2P1.fnoc

```
      SUBROUTINE    RRPSM(NCDS, NCDP, NCDCN, NDBIN1, NDBIN2,
     1                    WS, WD, MM, MU, IPXU, IPOFF, Z0M, JMAX, AA,
     2                    RHOUW, PP, QQ, RR, BIN,
     3                    LTEST, NOUT, CNT)
      IMPLICIT      REAL*8 (A - H, O - Z)
      LOGICAL       LTEST
      DIMENSION     WS(NCDS,NCDS,NCDP), WD(NCDS,NCDS,NCDP),
     1              MU(NCDS),
     2              PP(NCDS,NCDS,NCDCN), QQ(NCDS,NCDS,NCDCN),
     3              RR(NCDS,NCDS), BIN(NDBIN1,NDBIN2),
     4              CNT(NCDS,NCDS,NCDCN)
      CHARACTER*6   NAME / 'RRPSM ' /
      CHARACTER*60  VERS
     */ ' 4.2P1, 95/10/13 -M_4.2 C$DIR 122                          ' /
      PARAMETER     (ZERO = 0.D0, ONE = 1.D0, TWO = 2.D0)
      PARAMETER     (NPDS = 49, NPDCN = 22)
      DIMENSION     CNTX(NPDS,NPDCN), RRX(NPDS,NPDS,NPDCN,NPDCN)
      LOGICAL       LCALL0
      DATA          LCALL0 / .TRUE. /
      IF (LCALL0) WRITE (NOUT,4000) NAME, VERS
      IF (MM      .GT. NCDS)       CALL ERR2(NAME, 1, 1, NOUT)
      IF (IPXU    .GT. NCDP)       CALL ERR2(NAME, 2, 1, NOUT)
      IF (JMAX+1  .GT. NCDCN)      CALL ERR2(NAME, 3, 1, NOUT)
      IF (ABS(AA) .LT. CDXTOL())   CALL ERR2(NAME, 4, 1, NOUT)
      IF (NPDS    .NE. NCDS)       CALL ERR2(NAME, 5, 1, NOUT)
      IF (NPDCN   .NE. NCDCN)      CALL ERR2(NAME, 6, 1, NOUT)
      IF (LCALL0 .AND. IPOFF .GT. 2) CALL ERR2(NAME, 11, 0, NOUT)
      IF (LCALL0 .AND. JMAX .LT. 2) CALL ERR2(NAME, 21, 0, NOUT)
      IF (LCALL0) LCALL0 = .FALSE.
      IF (JMAX .LT. 2) RETURN
      IF (RHOUW .GE. ZERO) ZUW = TWO * AA * RHOUW
      IF (RHOUW .LT. ZERO) ZUW = -RHOUW
      JMAX1 = JMAX + 1
      RHO0  = Z0M / (TWO * AA)
C$DIR NO_PARALLEL
      DO  68  M1 = 1,JMAX
          JR     = M1 - 2
          Z0MIR2 = ONE / (Z0M ** (JR + 2))
          ZZZA = BIN(-1+3,JR+1+1) * Z0MIR2
C$DIR NO_PARALLEL
          DO  22  I = 1,MM
C$DIR NO_PARALLEL
              DO  20  J = 1,MM
                  PP(J,I,M1) = ZERO
   20         CONTINUE
   22     CONTINUE
C$DIR NO_PARALLEL
          DO  24  I = 1,MM
              PP(I,I,M1) = ZZZA
   24     CONTINUE
          IF (Z0M .LE. ZUW) THEN
              IPXL = JR + 1 + IPOFF
              IF (IPXL .LE. IPXU) THEN
                  ZZZB0 = Z0MIR2 * RHO0 ** (IPXL - IPOFF + 1)
C$DIR NO_PARALLEL
                  DO  34  IPX = IPXL,IPXU
                      IP = IPX - IPOFF
                      ZZZB = - BIN(IP+3,JR+1+1) * ZZZB0
C$DIR NO_PARALLEL
                      DO  32  I = 1,MM
C$DIR NO_PARALLEL
                          DO  30  J = 1,MM
                              PP(J,I,M1) = PP(J,I,M1)
     1                                   + ZZZB * WD(J,I,IPX)
   30                     CONTINUE
   32                 CONTINUE
                      ZZZB0 = ZZZB0 * RHO0
   34             CONTINUE
```

```
              ENDIF
            ENDIF
C$DIR NO_PARALLEL
            DO  42  I = 1,MM
C$DIR NO_PARALLEL
                DO  40  J = 1,MM
                    QQ(J,I,M1) = ZERO
   40           CONTINUE
   42       CONTINUE
            IF (JR .LT. 0) GO TO  66
                IF (JR .EQ. 0) THEN
                    ZZZC = -0.25D0
C$DIR NO_PARALLEL
                    DO  50  I = 1,MM
                        QQ(I,I,M1) = ZZZC
   50               CONTINUE
                ENDIF
                IF (JR .GE. 0) THEN
                    ZZZD = -BIN(-2+3,JR+1) * Z0MIR2
C$DIR NO_PARALLEL
                    DO  52  I = 1,MM
                        QQ(I,I,M1) = QQ(I,I,M1) + ZZZD * MU(I) ** 2
   52               CONTINUE
                ENDIF
            IF (Z0M .GT. ZUW) GO TO  66
                IPXL = -1 + IPOFF
                IF (IPXL .GT. IPXU) CALL ERR2(NAME, 31, 1, NOUT)
                ZZZE0 = Z0MIR2 * RHO0 ** (IPXL - IPOFF + 2)
C$DIR NO_PARALLEL
                DO  65  IPX = IPXL,IPXU
                    IP = IPX - IPOFF
                    IF (IP .GE. 0  .AND.  JR .GT. IP) GO TO  64
                        ZZZE = - BIN(IP+3,JR+1) * ZZZE0
C$DIR NO_PARALLEL
                        DO  62  I = 1,MM
C$DIR NO_PARALLEL
                            DO  60  J = 1,MM
                                QQ(J,I,M1) = QQ(J,I,M1)
     &                                       + ZZZE * WS(J,I,IPX)
   60                       CONTINUE
   62                   CONTINUE
   64               CONTINUE
                    ZZZE0 = ZZZE0 * RHO0
   65           CONTINUE
   66       CONTINUE
   68 CONTINUE
C$DIR NO_PARALLEL
      DO  206  N1 = 3,JMAX1
            JP  = N1 - 1
            JP1 = JP - 1
C$DIR NO_PARALLEL
            DO  204  M1 = 1,JP1
                JR   = M1 - 2
                ZZZP = JP - JR - 2
C$DIR NO_PARALLEL
                DO  202  K = 1,MM
C$DIR NO_PARALLEL
                    DO  200  J = 1,MM
                        RRX(J,K,M1,N1) = ZZZP * PP(J,K,M1) + QQ(J,K,M1)
  200               CONTINUE
  202           CONTINUE
  204       CONTINUE
  206 CONTINUE
C$DIR LOOP_PARALLEL
C$DIR LOOP_PRIVATE(CNTX,J,N1,JP,JP1,M1,JR,K,ZZZ71)
      DO  122  I = 1,MM
C$DIR NO_PARALLEL
            DO  70  J = 1,MM
                CNTX(J,2) = CNT(I,J,2)
   70       CONTINUE
C$DIR NO_PARALLEL
            DO  120  N1 = 3,JMAX1
```

```
              JP = N1 - 1
C$DIR NO_PARALLEL
              DO  72  J = 1,MM
                  CNTX(J,N1) = ZERO
   72         CONTINUE
              JP1 = JP - 1
C$DIR NO_PARALLEL
              DO  90  M1 = 1,JP1
                  JR = M1 - 2
C$DIR NO_PARALLEL
                  DO  86  K = 1,MM
C$DIR NO_PARALLEL
                      DO  84  J = 1,MM
                          CNTX(J,N1) = CNTX(J,N1)
     1                        + RRX(J,K,M1,N1) * CNTX(K,N1-JR-2)
   84                 CONTINUE
   86             CONTINUE
   90         CONTINUE
              M1 = JP
C$DIR NO_PARALLEL
              DO  92  J = 1,MM
                  CNTX(J,N1) = CNTX(J,N1) + QQ(J,I,M1)
   92         CONTINUE
              ZZZ71 = - ONE / DFLOAT(JP * (JP - 1))
C$DIR NO_PARALLEL
              DO  100  J = 1,MM
                  CNTX(J,N1) = ZZZ71 * CNTX(J,N1)
  100         CONTINUE
C$DIR NO_PARALLEL
              DO  110  J = 1,MM
                  CNT(I,J,N1) = CNTX(J,N1)
  110         CONTINUE
  120     CONTINUE
  122 CONTINUE
      IF (.NOT.LTEST) RETURN
      WRITE (NOUT,4100) NAME, Z0M
C$DIR NO_PARALLEL
      DO  132  K = 1,JMAX1
          WRITE (NOUT,4110) K
C$DIR NO_PARALLEL
          DO  130  I = 1,MM
              WRITE (NOUT,4120) (CNT(I,J,K), J = 1,MM)
  130     CONTINUE
  132 CONTINUE
      RETURN
 4000 FORMAT (
     A  2H -, A6, 1H-, A60)
 4100 FORMAT (//, 1H , 80(1H-), /,
     A  1H , 1H-, 2A4, 1H-, 4HZ0M , F12.6)
 4110 FORMAT (/,
     A  1H , 4H  K , I4)
 4120 FORMAT (
     A (1H , 8X, 4F16.8))
      END
```

File: rrpsm_4.1PV1.fnoc

```
      SUBROUTINE    RRPSM(NCDS, NCDP, NCDCN, NDBIN1, NDBIN2,
     1                    WS, WD, MM, MU, IPXU, IPOFF, Z0M, JMAX, AA,
     2                    RHOUW, PP, QQ, RR, BIN,
     3                    LTEST, NOUT, CNT)
      IMPLICIT      REAL*8 (A - H, O - Z)
      LOGICAL       LTEST
      DIMENSION     WS(NCDS,NCDS,NCDP), WD(NCDS,NCDS,NCDP),
     1              MU(NCDS),
     2              PP(NCDS,NCDS,NCDCN), QQ(NCDS,NCDS,NCDCN),
     3              RR(NCDS,NCDS), BIN(NDBIN1,NDBIN2),
     4              CNT(NCDS,NCDS,NCDCN)
      CHARACTER*6   NAME / 'RRPSM ' /
      CHARACTER*60  VERS
     */ ' 4.1PV1, 95/10/20 -M_4.1P1 (DGEMV)                          ' /
      PARAMETER     (ZERO = 0.D0, ONE = 1.D0, TWO = 2.D0)
      PARAMETER     (NPDS = 49, NPDCN = 22)
      DIMENSION     CNTX(NPDS,NPDCN), CNTY(NPDS,NPDCN)
      LOGICAL       LCALL0
      DATA          LCALL0 / .TRUE. /
      IF (LCALL0) WRITE (NOUT,4000) NAME, VERS
      IF (MM      .GT. NCDS)       CALL ERR2(NAME, 1, 1, NOUT)
      IF (IPXU    .GT. NCDP)       CALL ERR2(NAME, 2, 1, NOUT)
      IF (JMAX+1  .GT. NCDCN)      CALL ERR2(NAME, 3, 1, NOUT)
      IF (ABS(AA) .LT. CDXTOL())   CALL ERR2(NAME, 4, 1, NOUT)
      IF (NPDS    .NE. NCDS)       CALL ERR2(NAME, 5, 1, NOUT)
      IF (NPDCN   .NE. NCDCN)      CALL ERR2(NAME, 6, 1, NOUT)
      IF (LCALL0 .AND. IPOFF .GT. 2) CALL ERR2(NAME, 11, 0, NOUT)
      IF (LCALL0 .AND. JMAX .LT. 2) CALL ERR2(NAME, 21, 0, NOUT)
      IF (LCALL0) LCALL0 = .FALSE.
      IF (JMAX .LT. 2) RETURN
      IF (RHOUW .GE. ZERO) ZUW = TWO * AA * RHOUW
      IF (RHOUW .LT. ZERO) ZUW = -RHOUW
      JMAX1 = JMAX + 1
      RHO0  = Z0M / (TWO * AA)
C$DIR NO_PARALLEL
      DO  68  M1 = 1,JMAX
          JR     = M1 - 2
          Z0MIR2 = ONE / (Z0M ** (JR + 2))
          ZZZA = BIN(-1+3,JR+1+1) * Z0MIR2
C$DIR NO_PARALLEL
          DO  22  I = 1,MM
C$DIR NO_PARALLEL
              DO  20  J = 1,MM
                  PP(J,I,M1) = ZERO
   20         CONTINUE
   22     CONTINUE
C$DIR NO_PARALLEL
          DO  24  I = 1,MM
              PP(I,I,M1) = ZZZA
   24     CONTINUE
          IF (Z0M .LE. ZUW) THEN
              IPXL = JR + 1 + IPOFF
              IF (IPXL .LE. IPXU) THEN
                  ZZZB0 = Z0MIR2 * RHO0 ** (IPXL - IPOFF + 1)
C$DIR NO_PARALLEL
                  DO  34  IPX = IPXL,IPXU
                      IP = IPX - IPOFF
                      ZZZB = - BIN(IP+3,JR+1+1) * ZZZB0
C$DIR NO_PARALLEL
                      DO  32  I = 1,MM
C$DIR NO_PARALLEL
                          DO  30  J = 1,MM
                              PP(J,I,M1) = PP(J,I,M1)
     1                                   + ZZZB * WD(J,I,IPX)
   30                     CONTINUE
   32                 CONTINUE
                      ZZZB0 = ZZZB0 * RHO0
   34             CONTINUE
```

```
            ENDIF
          ENDIF
C$DIR NO_PARALLEL
          DO  42  I = 1,MM
C$DIR NO_PARALLEL
              DO  40  J = 1,MM
                  QQ(J,I,M1) = ZERO
   40         CONTINUE
   42     CONTINUE
          IF (JR .LT. 0) GO TO  66
              IF (JR .EQ. 0) THEN
                  ZZZC = -0.25D0
C$DIR NO_PARALLEL
                  DO  50  I = 1,MM
                      QQ(I,I,M1) = ZZZC
   50             CONTINUE
              ENDIF
              IF (JR .GE. 0) THEN
                  ZZZD = -BIN(-2+3,JR+1) * Z0MIR2
C$DIR NO_PARALLEL
                  DO  52  I = 1,MM
                      QQ(I,I,M1) = QQ(I,I,M1) + ZZZD * MU(I) ** 2
   52             CONTINUE
              ENDIF
          IF (Z0M .GT. ZUW) GO TO  66
              IPXL = -1 + IPOFF
              IF (IPXL .GT. IPXU) CALL ERR2(NAME, 31, 1, NOUT)
              ZZZE0 = Z0MIR2 * RHO0 ** (IPXL - IPOFF + 2)
C$DIR NO_PARALLEL
              DO  65  IPX = IPXL,IPXU
                  IP = IPX - IPOFF
                  IF (IP .GE. 0  .AND.  JR .GT. IP) GO TO  64
                      ZZZE = - BIN(IP+3,JR+1) * ZZZE0
C$DIR NO_PARALLEL
                      DO  62  I = 1,MM
C$DIR NO_PARALLEL
                          DO  60  J = 1,MM
                              QQ(J,I,M1) = QQ(J,I,M1)
     &                                     + ZZZE * WS(J,I,IPX)
   60                     CONTINUE
   62                 CONTINUE
   64             CONTINUE
                  ZZZE0 = ZZZE0 * RHO0
   65         CONTINUE
   66     CONTINUE
   68 CONTINUE
C$DIR LOOP_PARALLEL
C$DIR LOOP_PRIVATE(CNTX,CNTY,J,N1,JP,JP1,M1,JR,K,ZZZ71)
      DO  122  I = 1,MM
C$DIR NO_PARALLEL
          DO  70  J = 1,MM
              CNTX(J,2) = CNT(I,J,2)
   70     CONTINUE
C$DIR NO_PARALLEL
          DO  71  J = 1,MM
              CNTY(J,2) = CNTX(J,2)
   71     CONTINUE
C$DIR NO_PARALLEL
          DO  120  N1 = 3,JMAX1
              JP = N1 - 1
C$DIR NO_PARALLEL
              DO  72  J = 1,MM
                  CNTX(J,N1) = ZERO
   72         CONTINUE
              JP1 = JP - 1
C$DIR NO_PARALLEL
              DO  90  M1 = 1,JP1
                  JR = M1 - 2
                  CALL DGEMV('N', MM, MM, ONE, PP(1,1,M1), NCDS,
     &                       CNTY(1,N1-JR-2), 1,
     &                       ONE, CNTX(1,N1), 1)
                  CALL DGEMV('N', MM, MM, ONE, QQ(1,1,M1), NCDS,
```

```
     &                       CNTX(1,N1-JR-2), 1,
     &                       ONE, CNTX(1,N1), 1)
   90         CONTINUE
              M1 = JP
C$DIR NO_PARALLEL
              DO  92  J = 1,MM
                  CNTX(J,N1) = CNTX(J,N1) + QQ(J,I,M1)
   92         CONTINUE
              ZZZ71 = - ONE / DFLOAT(JP * (JP - 1))
C$DIR NO_PARALLEL
              DO  100  J = 1,MM
                  CNTX(J,N1) = ZZZ71 * CNTX(J,N1)
  100         CONTINUE
C$DIR NO_PARALLEL
              DO  102  J = 1,MM
                  CNTY(J,N1) = (N1 - 1) * CNTX(J,N1)
  102         CONTINUE
C$DIR NO_PARALLEL
              DO  110  J = 1,MM
                  CNT(I,J,N1) = CNTX(J,N1)
  110         CONTINUE
  120     CONTINUE
  122 CONTINUE
      IF (.NOT.LTEST) RETURN
      WRITE (NOUT,4100) NAME, Z0M
C$DIR NO_PARALLEL
      DO  132  K = 1,JMAX1
          WRITE (NOUT,4110) K
C$DIR NO_PARALLEL
          DO  130  I = 1,MM
              WRITE (NOUT,4120) (CNT(I,J,K), J = 1,MM)
  130     CONTINUE
  132 CONTINUE
      RETURN
 4000 FORMAT (
     A  2H -, A6, 1H-, A60)
 4100 FORMAT (//, 1H , 80(1H-), /,
     A  1H , 1H-,  A6, 1H-, 4HZ0M , F12.6)
 4110 FORMAT (/,
     A  1H , 4H  K , I4)
 4120 FORMAT (
     A (1H , 8X, 4F16.8))
 6000 FORMAT (
     A  1H , 1H-,  A6, 1H-, 4H   I, I8, 8H  THREAD, I8)
      END
```