RMPROP On-Line Manual

Program version: 2.1.2
Version Date: December 21, 1994
Date of this manual update: April 5, 1995

Melissa S. Reeves, Michael J. Unekis, David W. Schwenke, Nancy M. Harvey,
and Donald G. Truhlar*
Department of Chemistry, Chemical Physics Program, and Supercomputer Institute
University of Minnesota, Minneapolis, MN 55455
and NASA Ames Research Center, Moffett Field, CA 94035

*
University of Minnesota

+
NASA Ames Research Center

CONTENTS

(Table of Contents is continued on page ii.)

1. ABSTRACT

RMPROP is a computer program for solving the quantum mechanical scattering problem for single-arrangement collisions using the R matrix propagation algorithm. The program yields the scattering matrix for a particular total energy and total angular momentum. Special features of the algorithm are that it vectorizes very efficiently, closed channels are treated without numerical difficulty, the integration stepsize can be large in regions where the interaction and centrifugal potentials are slowly varying functions of the propagation coordinate, and in many cases much of the computational effort can be reused for various types of calculations. RMPROP can solve the scattering problem subject to homogeneous or nonhomogeneous boundary conditions at small values of the scattering coordinate (the scattering boundary conditions are always nonhomogeneous at large values of the scattering coordinate).

Version 2.1.2 executes in single-processor mode, except for certain matrix operations where multitasking may be invoked, and it is optimized for a vector pipeline computer. It is distributed with a test suite and this on-line manual.

1.1 ACKNOWLEDGMENT

Acknowledgment of support: Development of this program was supported in part by the National Science Foundation, the Minnesota Supercomputer Institute, and Cray Research, Inc.

2. GENERAL REFERENCES

1. J. C. Light and R. B. Walker, "An R matrix approach to the solution of coupled equations for atom-molecule reactive scattering", J. Chem. Phys., Vol. 65, 1976, pp. 4272-4282.

2. N. M. Harvey, "New Methods for Quantum Mechanical Calculations of Inelastic Atom-Molecule Collisions and Electron Scattering", Ph.D. dissertation, University of Minnesota, Minneapolis, 1979. xi+365 pp.

3. D. G. Truhlar, N. M. Harvey, K. Onda, and M. A. Brandt, "Applications of Close Coupling Algorithms to Electron-Atom, Electron-Molecule, and Atom-Molecule Scattering", in "Algorithms and Computer Codes for Atomic and Molecular Scattering Theory", Vol. 1, edited by L. Thomas, National Resource for Computation in Chemistry, Lawrence Berkeley Laboratory, Berkeley, CA, 1979, pp. 220-289.

4. D. W. Schwenke and D. G. Truhlar, "Large-Scale Quantum Mechanical Scattering Calculations on Vector Computers", in "Supercomputer Applications", edited by R. W. Numrich, Plenum, New York, 1985, pp. 215-254.

5. M. J. Unekis, D. W. Schwenke, N. M. Harvey, and D. G. Truhlar, "RMPROP: A Computer Program for Quantum Mechanical Close Coupling Calculations for Inelastic Collisions", in "Modern Techniques in Computational Chemistry: MOTECC-91", edited by E. Clementi, ESCOM, Leiden, 1991, pp. 749-772.

6. M. J. Unekis, D. W. Schwenke, N. M. Harvey, and D. G. Truhlar, "RMPROP-Version 2: A Computer Program for Quantum Mechanical Close Coupling Calculations for Inelastic Collisions", in "Modern Techniques in Computational Chemistry: METECC-94 Vol. C", edited by E. Clementi, STEF, Cagliara, Italy, 1993, pp. 1-46.

For information on the alternative real symmetric matrix eigensystem routine, SUBROUTINE EVVRSP, see:

7. S. T. Elbert, "Extracting More than a Few Eigenvectors from a Dense Real*8 Symmetric Matrix: Optimal Algorithms versus the Architectural Constraints of the FPS-X64", Theor. Chim. Acta, Vol. 71, 1987, pp.169-186.

For information on the Level-1 BLAS, see:

8. Dodson, D., and Lewis, J., "Issues Relating to Extension of the Basic Linear Algebra Subprograms", ACM SIGNUM Newsletter 20, 1985, pp. 2-18.

For information on the Level-2 BLAS, see:

9. Dongarra J. J., Du Croz J. J., Hammarling S., and Hanson R. J., "An extended set of Fortran Basic Linear Algebra Subprograms", Technical Memoranda Nos. 41 (revision 3) and 81, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439, USA.

10. NAG Technical Reports TR3/87 and TR4/87, Numerical Algorithms Group Ltd., NAG Central Office, 256 Banbury Road, Oxford OX2 7DE, UK, and Numerical Algorithms Group Inc., 1101 31st Street, Suite 100, Downers Grove, Illinois 60515-1263, USA.

For more information on the Level-3 BLAS, see:

11. J. J. Dongarra, J. J. Du Croz, I. Duff, and S. Hammarling, "A set of Level 3 Basic Linear Algebra Subprograms. Technical Memorandum No. 88 (Revision 1)", Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439, USA.

For more information on LAPACK, see:

12. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, "LAPACK User’s Guide," SIAM, Philadelphia, 1992.

3. DISTRIBUTION

RMPROP is copyrighted and is distributed by the authors and by Modern Techniques in Computational Chemistry Club, a non-profit educational organization, Enrico Clementi, President.

Users are requested not to distribute any portion of the RMPROP software or any derivative works based upon the RMPROP software without prior authorization by the authors.

Use of the RMPROP program should be appropriately acknowledged in publications, e. g., by the following two references.

M. S. Reeves, M. J. Unekis, D. W. Schwenke, N. M. Harvey, and D. G. Truhlar, RMPROP-Version 2.1.2, University of Minnesota, Minneapolis, 1994.

M. J. Unekis, D. W. Schwenke, N. M. Harvey, and D. G. Truhlar, "RMPROP-Version 2: A Computer Program for Quantum Mechanical Close Coupling Calculations for Inelastic Collisions", in "Modern Techniques in Computational Chemistry: METECC-94 Vol. C", edited by E. Clementi, STEF, Cagliara, Italy, 1993, pp. 1-46.

4. PROGRAM DOCUMENTATION AND HISTORY

The documentation for this code consists of two parts: this on-line manual and Ref. 6 of Section 2 of this manual. The other general references of Section 2 also provide important background information. See Section 7 of this on-line manual for a section-by-section discussion of the contents of this manual.

4.1 VERSION HISTORY

Version	Date	Summary of Features
1.0	6/1/91	Refer to MOTECC-91
2.0	9/1/93	Refer to METECC-94 (Note: Version 2.0 was never distributed.)
2.1	8/9/94	1. Meaning of EPSRED changed. 2. Fixed bug with closed channels in asymptotic analysis (only in 2.0). 3. Test suite expanded and altered. 4. Several other bug, input, and documentation fixes.
2.1.1	8/30/94	1. One-line change made in flambda.f and jsymbol.f 2. Function FCOEF5 enabled (had been disabled in 2.1)
2.1.2	12/21/94	1. IBM 3090 no longer supported by distribution package. 2. Manual was improved.

5. INTRODUCTION

RMPROP is a general program utilizing the R matrix propagation algorithm for the solution of the quantum mechanical close coupling equations as applied to atomic and molecular collisions to obtain the scattering matrix. It has been used for electron-atom scattering, three-dimensional electron-diatom scattering, collinear atom-diatom scattering, three-dimensional atom-diatom scattering, and three-dimensional diatom-diatom scattering. Earlier versions have also been applied to collinear diatom-diatom scattering and to collisions of electrons, atoms, or rigid rotators with rigid rotators, and the code is general enough to be applied to any single-arrangement close coupling calculations in which the kinetic energy operator does not couple different channels. Since the error per step of the integration is determined by the rate of change of the potential rather than the rate of oscillation of the wavefunction, the program is especially well suited for treating heavy-body collisions or systems with long-range potentials.

In order to apply this program to a system not included in the test suite, the user must supply either subroutines to compute the coupling matrix elements or a subroutine which gives the interaction potential as a function of nuclear coordinates. Note however that the distributed version of RMPROP can only be interfaced to interaction potential subroutines for the systems (atom-diatom and diatom-diatom) for which routines are included to compute the coupling matrix elements. Depending on the system, a significant amount of resources can be spent in potential energy subroutines, so they should be optimized as much as possible. The requirements for user-supplied subroutines for coupling matrix elements are given in Section 10 of this manual, while the requirements for the user-supplied interaction potential are detailed in Section 11.

5.1 The distribution package

RMPROP Version 2.1.2 is distributed as a directory system with the following structure:

```
RMPROP2.1.2
          runtest.csh
          rmprop.doc
          script/
          src/
          test/
```

Executable file runtest.csh is a C shell script which will run elements of the test suite.
File rmprop.doc contains this on-line manual.
Subdirectory script contains all the scripts used to run the test suite elements and the makefile for the code.
Subdirectory src contains all the source code for all the versions of RMPROP 2.1.2.
Subdirectory test contains the input files, copies of the short output, and the include files for the test suite elements. The short output files contained here can be compared against results obtained on the user’s machine.

5.2 Running RMPROP 2.1.2: An overview

A. The Test Suite. The distribution package includes 16 test runs, with all the necessary input decks, interaction potential functions, potential matrix element routines, and shell scripts to run them. These are discussed in more detail in Section 19. To access the test suite, type

```
runtest.csh
```

in the top directory of the distribution package. You will be queried about your choice of test suite elements. Then the script will attempt to determine what type of machine you are running on and choose the appropriate machine-specific codes and compiler commands. If RMPROP does not recognize your machine type, you may need to supply certain codes for timing and then alter the scripts in the distribution package to run on your machine (see Section 6.1 of the manual for details). If RMPROP recognizes your machine (or after you have performed the necessary tasks), the scripts will compile the portable version of RMPROP and run the test suite element. The output for the test suite run will be in the top directory, labeled {testname}.out. A short output file, labeled {testname}.15 can be compared to the distributed copy of this file in subdirectory test.

CAUTION: RMPROP Version 2.1.2 uses a makefile to compile the codes, but the makefile does not check to see if the include files have been updated. If you wish to run different test suite elements, you must remove the object files created between the test runs. This feature will be altered in the next version of RMPROP.

B. User-defined calculations. To run a calculation with a potential not distributed with RMPROP, the user should examine Sections 10 and 11 for the necessary coding requirements. When utilizing the modular potential matrix routines described in Section 11, the input file fort.4 must be supplied. It is suggested that the user copy and modify one of the distributed copies given in subdirectory test. For any calculation not in the test suite, the user must supply the main input file fort.5. Again, it is suggested that the user modify one of the distributed copies of this file, given in subdirectory test. Complete documentation of the two input files fort.4 and fort.5 is given in Sections 11 and 12, respectively. For a user-defined problem, the include file parameter.inc must be supplied for dimensioning of the large arrays in RMPROP. This file is discussed in detail in Section 15. Utilization of the modular potential matrix routines may also require that the user provide the file expand.inc. The form of this file is described in detail in Section 11. The scripts for the test suite in subdirectory script serve the purpose of executing test suite elements as well as providing templates for users to run their own problems.

5.3 OVERVIEW OF MATRIX UTILITIES

The most intensive computations in RMPROP are matrix-matrix multiplication, solving linear systems, solving eigenvalue-eigenvector problems, and performing matrix inversions. These operations are common matrix operations, and we have selected routines from the BLAS and LAPACK libraries to perform them in RMPROP. The exception to this is the eigensystem analysis, which is done by modified versions of EVVRSP (see reference 7 in Section 2), or on the Cray computers by RS (originally from EISPACK, but modified in the Cray SCILIB library).

Many computers have optimized versions of the BLAS and LAPACK routines with the same calling sequence; linking to these machine-optimized versions rather than the FORTRAN source versions in the RMPROP distribution package will probably provide significant speed-up of RMPROP. Here we give a brief overview of the BLAS and LAPACK libraries.

The BLAS libraries contain Basic Linear Algebra Subroutines which perform vector-vector (Level 1), matrix-vector (Level 2), and matrix-matrix (Level 3) linear algebra operations. For background on these routines, refer to the references 8-11 in Section 2. The specific routines used in RMPROP are given in Section 17.

Matrix inversions and linear system solutions have been taken from LAPACK. For complete information on LAPACK, refer to reference 12 in Section 2. The specific routines used in RMPROP are given in Section 17.

6. COMPUTER AND OPERATING SYSTEM

The program RMPROP is a very portable code, and it has been tested on a number of different computers. We expect that any UNIX computer with a FORTRAN compiler should have no difficulty running RMPROP as distributed, except for minor incompatibilities in machine-specific subroutines described in the next subsection. For users interested in high performance, we have also included in our distribution package the code to run versions highly optimized for an IBM RS-6000, Cray-2, and Cray Y-MP C90 (hereafter referred to as a Cray C90). These optimized versions can be accessed by modifying the file Makefile.mak, as detailed in sections 6.5 and 6.7. In the following section we will detail the differences among these versions, as well as our recommendation for specific versions on specific machines.

RMPROP is written in FORTRAN 77. The code is explicitly written to run in 64-bit precision, i. e., REAL*8, using DOUBLE PRECISION declarations and intrinsic functions. The explicit use of double precision coding facilitates usage on machines (e. g., the IBM RS/6000 workstation or the DEC Alpha) where single precision is 32 bits. On Cray computers, single precision arithmetic is 64 bits, but this may be adjusted for by the use of compiler options; in particular use of the -dp compiler option ("disable precision") on Cray computers causes the code to execute in REAL*8 precision on a Cray computer as well.

RMPROP is intended for use with static memory allocation. The compiler options in the distribution package explicitly choose static memory allocation, and users of other computers should be aware that this is necessary.

The code has been well optimized for a Cray vector pipeline computer. This version has been successfully tested on a Cray-2 and Cray C90 with the UNICOS operating system, version 6.1, and on an IBM RS/6000 Model 550 computer with the AIX operating system, versions 3.1.5 and 3.2.1. Version 2.0 was also successfully tested on a Cray X-MP with UNICOS operating system version 6.1, Sun SPARCStation IPX, using SunOS version 4.1.2, and on a DEC 3000/500 ("Alpha") workstation, using the OSF/1 version T1.2-2 operating system.

6.1 MACHINE-SPECIFIC CODES

There are a number of subroutines which cannot be standardized from machine to machine, specifically those subroutines needed to determine machine epsilon and date and timing routines. We have placed most of these codes in files which begin with the suffix "sdr." These files are

File	Machine
sdralpha.f	DEC 3000/500AXP OSF/1
sdrcray.f	Cray-2, Cray C90
sdrrs6000.f, dateclock.c	IBM RS/6000
sdrsun.f	Sun

It should be noted that the individual shell scripts which run the elements of the test suite automatically choose the appropriate file(s) from the above list. The machine is identified by either of the commands

"'uname'"
or "'uname -m'"

where the quotes are present in the shell script.

If the machine and operating system are not among those recognized by RMPROP, the user must supply routines which perform the tasks in the sdr files and also modify the shell scripts for the test suite. The following subroutines must be supplied:

SUBROUTINE DATTIM(IO,TIMDAT): This subroutine should call routines on the local machine which will return the date and time. IO is an integer variable containing the fortran unit to which the date and time should be written. TIMDAT is a CHARACTER*80 variable which should contain the date and time as called in this routine.

SUBROUTINE SECOND(T): This subroutine should call routines on the local machine which give the cpu time since the start of execution of the code. T is a double precision variable which should, on return, contain the cputime in seconds since the code began execution.

6.2 SUMMARY OF VERSIONS OF RMPROP 2.1.2

The distribution package for RMPROP-Version 2.1.2 contains files for a portable version of the code, called version 2.1.2, which has been run successfully on a Cray C90, a Cray-2, and an IBM RS/6000 Model 550. The distribution package also contains several machine-specific versions optimized for a Cray-2, Cray C90, and IBM RS/6000.

In the rest of this manual, the portable version, which may be used on all machines, is called 2.1.2, the Cray-2 version with RS is called 2.1.2cs, the Cray-2 version with EVCRSP is called 2.1.2ce, the Cray C90 version with RS is called 2.1.2xs, the Cray C90 version with EVCRSP is called 2.1.2xe, and the IBM RS/6000 version (which includes EVIRSP) is called 2.1.2i.

The differences among these machine-specific versions lie in the choice of subroutines used for the N**3 matrix operations such as matrix- matrix multiplication, solution of linear systems, and eigensystem analysis. These operations are the most time-consuming parts of the propagation method, and choosing the optimal subroutine for a specific machine greatly enhances the speed of RMPROP on that machine.

The table below lists the different machine-specific versions of the program and indicates the major matrix utility routines used by each version. Detailed descriptions of each version of the program are given in Sections 6.3-6.8 below.

Version	2.1.2	2.1.2cs	2.1.2ce	2.1.2xs	2.1.2xe	2.1.2i
Machine	All	Cray-2	Cray-2	Cray C90	Cray C90	IBM RS/6000
Eigensystem Analysis Routine	EVDRSP	SCILIB RS	EVCRSP	SCILIB RS	EVCRSP	EVIRSP
Matrix Multiply Routine	FORTRAN source DGEMM	SCILIB MXMA	SCILIB MXMA	SCILIB SGEMM	SCILIB SGEMM	IBM library DGEMM
Linear Equation Solution	FORTRAN source DGETRS/ DGETRF	SCILIB MINV	SCILIB MINV	SCILIB LAPACK SGETRS/ SGETRF	SCILIB LAPACK SGETRS/ SGETRF	FORTRAN source DGETRS/ DGETRF

6.3 VERSION 2.1.2: PORTABLE VERSION

In order to run the portable version of the program, the list of object files to be compiled in the file "Makefile.mak" should read

```
OBJ= rmprop2.o exscat2.o gnscat2.o verp.o
```

The version with these files compiled and linked is called version 2.1.2. The distribution package is preset with this line in Makefile.mak. No changes need to be made.

The distribution version of RMPROP-Version 2.1.2 uses FORTRAN source subprograms from both the LAPACK library and the Basic Linear Algebra Subroutines (BLAS) to perform most matrix operations. In order to solve the complete eigensystem of a real symmetric matrix, the portable version of the program uses subroutine EV2RSP, which is a modification of subroutine EVVRSP written by Stephen T. Elbert of Iowa State University. This routine runs much more quickly than the FORTRAN 77 version of subroutine RS from EISPACK which had been used in RMPROP-Version 1.0. All of the matrix utility routines including EV2RSP are ANSI standard FORTRAN 77. These matrix utilities are discussed in more detail in Section 16 of this manual, and in Section 9 of the METECC-94 book chapter.

6.4 TIMING OF PORTABLE VERSION 2.1.2

The running time of RMPROP varies considerably, depending mainly on the system studied, the complexity of the potential, the stepsizes, the size of the basis sets used in the close coupling expansion of the Schroedinger equation, and the use of the inhomogeneous option (see Section {8.6} below). This option allows the program to solve the scattering problem for systems where some or all of the channel orbital angular momenta are zero, so that the collision partners approach arbitrarily close to one another. However this option is not necessary for most applications. The test runs below, except for test run ehydr1, were performed with the inhomogeneous option turned off, which reduces both the CPU and memory requirements of the program. However, all such elements of the test suite have been tested with the inhomogeneous option both ON and OFF, and final transition probabilities calculated using each option were in good agreement.

The table below gives sample timings for version 2.1.2 of the program on the Cray-2, Cray C90, and IBM RS/6000. The results in the table show that the Cray-2 is roughly comparable to the IBM RS/6000 for most runs of the test suite. There are two main reasons for this. The first is the small size of the elements of the test suite. The Cray vector pipeline architecture does not handle these small matrices efficiently, but the scalar architecture of the IBM RS/6000 does not require a minimum matrix size for optimum performance. The second reason is optimization of the program by the compilers. The FORTRAN source codes for matrix multiplication and eigensystem analysis included in the portable version of the program are not well-optimized by the Cray compiler, so it does not run as efficiently on the Cray vector pipeline architecture as the Cray SCILIB library routines. However the optimizing flag on the IBM RS/6000 compiler produces executable programs which run efficiently on the IBM scalar architecture. Sections 6.4 - 6.6 below show the results of using machine-optimized (i. e., system-specific FORTRAN and library routines) on the Cray and IBM architectures.

Test run no.	Test run name	CPU Time (seconds) for portable version 2.1.2		
		IBM RS/6000	Cray-2	Cray C90
1	hfhfstr1	278.2	317.9	114.2
2	hfhfstr2	452.2	434.1	152.8
3	hfhfstr3*	82.0	75.5	26.9
4	hfhfstr4	26.1	41.4	14.2
5	hfhfstr5	93.4	112.3	39.4
6	hehfr1	1.0	1.3	0.6
7	heh2cdr1	1.7	2.5	1.2
8	heh2cdr2	1.2	1.7	0.7
9	hfhfpbsr1	8.1	11.8	4.6
10	hfhfadr1	3.7	3.6	2.0
11	hfhfadr2*	1.8	2.2	1.1
12	atdir1	0.8	1.2	0.5
13	ehydr1	0.6	0.6	0.3
14	hei2r1	25.4	29.2	10.8
15	heh2tkr1	6.0	9.8	3.3
16	heh2tkr2	5.0	8.2	2.8

* (reads in restart information from previous run)

6.5 VERSIONS 2.1.2cs, 2.1.2ce, 2.1.2xs, 2.1.2xe: CRAY UNICOS VERSIONS

6.5.1 Version 2.1.2cs

On the Cray-2, optimized subroutines for matrix multiplication and eigensystem analysis are available in the Cray SCILIB library. Version 2.1.2cs makes full use of the SCILIB library by calling MXMA for matrix multiplication, MINV for linear equation solutions, and RS for eigensystem analysis. The use of these subroutines will greatly enhance performance of RMPROP on the Cray-2.

In order to use Version 2.1.2cs, the list of object files to be compiled in the file "Makefile.mak" should read

"OBJ= rmprop2.o exscat2.o gnscat2.o vercs.o"

6.5.2 Version 2.1.2ce

Also for use on the Cray-2 running UNICOS is version 2.1.2ce, which differs

from 2.1.2cs only in that it calls EV2RSP for eigensystem analysis instead of RS. For runs in the test suite included with this manual, the Cray SCILIB version of SUBROUTINE RS is the faster of the two. However, for larger runs, we have found that EVVRSP is more efficient. For example, for runs on the HF-HF system used in test runs 1-4 of this manual, EVCRSP gives comparable performance to SCILIB RS for matrices of dimension 236 x 236, and outperforms it by 36% for the same HF-HF system for matrices of dimension 1058 x 1058. See Section 17 of this manual for more information.

In order to run the program on the Cray-2 with the Cray SCILIB version of the matrix multiplication routine MXMA, and of linear solution routine MINV, and with the modified eigensystem analysis routine, EVCRSP, the list of object files to be compiled in the file "Makefile.mak" should read

"OBJ= rmprop2.o exscat2.o gnscat2.o verce.o" .

The version with these files compiled and linked is called version 2.1.2ce.

In order to run the program on the Cray C90 with the Cray SCILIB version of the matrix multiplication subroutine SGEMM and linear equation solution routines SGETRF/SGETRS, and with the Cray SCILIB version of the eigensystem analysis routine, RS, the list of object files to be compiled in the file "Makefile.mak" should read

"OBJ= rmprop2.o exscat2.o gnscat2.o verxs.o" .

The version with these files compiled and linked is called version 2.1.2xs.

In order to run the program on the Cray C90 with the Cray SCILIB version of the matrix multiplication routine SGEMM, and of linear solution routines SGETRF/SGETRS, and with the modified eigensystem analysis routine, EVCRSP, the list of object files to be compiled in the file "Makefile.mak" should read

"OBJ= rmprop2.o exscat2.o gnscat2.o verxe.o" .

The version with these files compiled and linked is called version 2.1.2xe.

6.6 TIMING OF CRAY UNICOS VERSIONS 2.1.2cs, 2.1.2ce, 2.1.2xs, 2.1.2xe

Below we list the timings for the test suite using the distributed version of the program with the SCILIB matrix multiplication routine MXMA and linear equation solver MINV, and with either the SCILIB eigensystem analysis routine RS, or the alternative eigensystem analysis routine EVCRSP:

TEST run no.	TEST run name	CPU Time (seconds) for Cray UNICOS versions: Cray-2 2.1.2cs	Cray-2 2.1.2ce	Cray C90 2.1.2xs	Cray C90 2.1.2xe
1	hfhfstr1	149.9	144.3	40.4	50.6
2	hfhfstr2	188.8	171.1	54.7	65.0
3	hfhfstr3*	27.4	26.1	7.9	10.2
4	hfhfstr4	32.7	33.5	10.8	11.2
5	hfhfstr5	62.0	64.9	9.3	22.4
6	hehfr1	0.8	0.7	0.4	0.4
7	heh2cdr1	5.7	4.6	2.6	1.0
8	heh2cdr2	1.2	0.8	0.3	0.4
9	hfhfpbsr1	8.0	5.7	2.4	2.5
10	hfhfadr1	4.9	2.8	1.7	1.7
11	hfhfadr2*	1.9	1.8	1.0	1.0
12	atdir1	1.0	0.6	0.2	0.2
13	ehydr1	0.3	0.3	0.2	0.2
14	hei2r1	13.8	12.8	4.4	5.0
15	heh2tkr1	8.3	8.3	2.7	2.8
16	heh2tkr2	6.8	6.8	2.2	2.3

* (reads in restart information from previous run)

6.7 VERSION 2.1.2i: IBM RS/6000 VERSION

On the IBM RS/6000, the matrix multiplication and linear equation solutions which are carried out by calls to FORTRAN 77 versions of the BLAS routines contained in module "matrix.f" may instead call the BLAS routines contained in the directory "/lib/libblas.a". The IBM RS/6000 version of the program also calls a version of Stephen Elbert’s eigensystem analysis routine, which has been optimized for the IBM by the process of rewriting important FORTRAN DO-LOOPs in terms of calls to the BLAS. This routine is called EVIRSP to distinguish it from the distribution version and the Cray-specific version of the subroutine. Finally, many of the FORTRAN DO LOOPs within RMPROP itself are replaced by calls to the BLAS. The use of the BLAS and of EVIRSP greatly improves performance on the IBM RS/6000.

It should be noted that although many of the BLAS routines (e. g. DGEMM) are present in "/lib/libblas.a" on the IBM RS/6000, the LAPACK routines (e. g. DGETRF/DGETRS) are not present with this directory, and so must be included as ANSI standard FORTRAN 77 modules. However, even in these cases, the total execution time of these subroutines may be decreased by allowing calls to the BLAS subroutines within these subprograms to be handled by the BLAS whenever possible.

In order to run the program on the IBM RS/6000 with the IBM version of the BLAS routines, and with the IBM-optimized version of Stephen Elbert’s eigensystem analysis routine, EVIRSP, the list of object files to be compiled in the shell script "Makefile.mak" should read

"OBJ= rmprop2.o exscat2.o gnscat2.o veri.o".

The version with these files compiled and linked is called version 2.1.2i.

6.8 TIMING OF IBM RS/6000 VERSION 2.1.2i

Below we list the timings for the test suite using the IBM RS/6000 version of the program with the LAPACK routines and the modified alternative eigensystem analysis routine EVIRSP:

Test run no.	Test run name	CPU Time (seconds) for version 2.1.2i
1	hfhfstr1	202.8
2	hfhfstr2	318.4
3	hfhfstr3*	56.9
4	hfhfstr4	26.0
5	hfhfstr5	71.2
6	hehfr1	1.2
7	heh2cdr1	1.9
8	heh2cdr2	1.0
9	hfhfpbsr1	7.8
10	hfhfadr1	3.5
11	hfhfadr2*	1.7
12	atdir1	0.8
13	ehydr1	0.7
14	hei2r1	19.8
15	heh2tkr1	6.0
16	heh2tkr2	5.0

* (reads in restart information from previous run)

7. ORGANIZATION OF THE MANUAL

The theoretical background for the dynamical calculations is described in a chapter by the authors of this program that has been published in "Modern Techniques in Computational Chemistry: METECC-94", edited by Enrico Clementi. Section 8 of details necessary to understand the input data. Section 9 gives a brief summary of the units used in the program. Section 10 describes the argument list for the subroutine which passes potential matrix elements to the main program when the matrix elements are known explicitly or can be calculated analytically from the interaction potential function directly. Section 11 describes the argument list for a user-supplied potential energy surface and subroutines required by energy surface. Section 12 lists input and output files used by the program. Section 13 gives the input to the main code from FORTRAN unit 5. Section 14 discusses array dimensions and memory considerations. Section 15 gives a list of the labeled common blocks used. Section 16 contains an alphabetical description of all the subroutines in the program, and a description of the system-dependent subroutines used to perform non-computational tasks. Section 17 discusses the matrix utilities and their subsidiary routines. Section 18 contains a summary of fatal error messages issued by the main program. Section 19 contains sample runs which illustrate various options allowed by the program; these may be useful in understanding the input or in testing the program. Section 20 lists the files in the program distribution. Section 21 is the bibliography, Section 22 contains known errata in the on-line manual for RMPROP-Version 1.0, and Section 23 contains known bugs in the FORTRAN program for RMPROP-Version 1.0.

7.1 MANUAL CONVENTIONS, BRACES, BRACKETS

Subprogram names, input variables, and variable names from the RMPROP program are in all capital letters. File names are in all lower case letters. References to the theoretical discussion in Section 8 are made by paragraph numbers given in curly brackets, (e.g., {8.3} refers to paragraph 3 in Section 8). References to equations from the METECC-94 chapter are made in brackets (e.g., [5] refers to Equation 5. Greek letters used as variables are always spelled out (i.e., alpha), and capitalized if a capital Greek letter is implied (i.e., Beta).

8. THEORETICAL BACKGROUND FOR DYNAMICAL CALCULATIONS

8.1 This section provides a brief overview of the dynamical theory underlying the calculations, with the emphasis on relating the theoretical formulation to the program input data. Thus input variable names are used directly in the discussion. The paragraphs are numbered to facilitate cross references. A complete list of input variables is given in Section 13 below. A discussion of manual conventions, braces, and brackets is given above in Section 7.1.

8.2 For concreteness we consider the collision of two molecules, A and B. The coordinate r is the distance between the centers of mass of A and B, and x is the collection of all other coordinates describing the system. It is assumed that a set of r-independent primitive orthonormal basis functions $X_n(x)$, $n = 1, \ldots, NDIM$, have been chosen, that these functions are eigenfunctions of the total angular momentum with eigenvalue JTOT(JTOT+1) in atomic units, and that the wavefunction has been scaled so that the full Hamiltonian has the form

$$H = -\frac{1}{2\,MUABC}\frac{d^2}{dr^2} + H^0(r,x) + scriptV(r,x) \qquad (1)$$

where

$$H^0(r,x)\,X_n(x) = \left(epsilon_n + \frac{L(n)\,(\,L(n)+1\,)}{2\,MUABC\,r^2}\right) X_n(x). \qquad (2)$$

Here $epsilon_n$ is the internal energy of eigenfunction X_n, and each L(n) is the orbital angular momentum for relative translational motion associated with the corresponding eigenfunction X_n. It is also assumed that the interaction between the collision partners, V(r,x), which is included in scriptV(r,x), decreases faster at large r than 1/r. Unlike RMPROP-Version 1.0, however, it is not required that V(r,x) become much larger than the scattering energy as r goes to zero. The program can now handle this case, as well as the opposite situation, by the use of the inhomogeneous option, governed by the input logical variable LR4MT {8.5, 8.6}. Each primitive function $X_n(x)$ defines what is called a channel.

8.3 Then the problem is to determine the coefficients in the large-r boundary conditions for the radial functions $f_{nn_0}(r)$ in the wavefunction expansion

$$\mathrm{Psi}_{n_0}(r, x) = \frac{1}{r} \sum_{n=1}^{\mathrm{NDIM}} X_n(x)\, f_{nn_0}(r). \qquad (3)$$

The large-r boundary conditions take the form

$$\lim_{r \to \infty} f_{nn_0}(r) = \delta_{nn_0}\, j_n(r) + n_n(r)\, K_{nn_0} \qquad (4)$$

where j_n and n_n are matching functions, and the K_{nn_0} are the unknown elements of the reactance matrix. The scattering matrix is determined from the reactance matrix by well known equations.

8.4 We proceed by dividing the propagation coordinate r into sectors with centers RABC(i). The width of sector i is H(i) (not to be confused with the H_0 or the H_0 of {8.2}). The center of the first sector is RSTART, and propagation proceeds according to either of two options selected by the user by a variable LR4MT {8.5, 8.6} set in FORTRAN unit 5. For i greater than 2, the width of the next sector is determined automatically. First a trial width is determined using epsilon = EPSA, EPSB, or EPSC and h_{max} = HMAXA, HMAXB, or HMAXC in [47]. The trial width is set to the minimum of the width determined using the appropriate epsilon and h_{max}. The trial width is then compared to the input parameter HMINA, HMINB, or HMINC, and the final stepsize is the maximum of the trial width and the HMINA/HMINB/HMINC parameter. The choice between HMINA, HMINB, or HMINC for the minimum value and between HMAXA, HMAXB, or HMAXC for the maximum value is made as follows: HMINA and HMAXA are used if r is less than REPSA, HMINC and HMAXC are used if r is greater than REPSB, and HMINB and HMAXB are used otherwise. This width is used unless one of the NCUSP values of RCUSP lies within this distance of RABC(i). In that case, the width is determined by setting RABC(i+1) to the next value of RCUSP. The propagation continues either until the scattering matrix converges with respect to r, until a value of RABC(i) + H(i)/2 exceeds the limit RMAX, or until the number of sectors exceeds the dimension NSECT set in the program.

8.5 The option set by the variable LR4MT is known as the (in)homogeneous option. If LR4MT = .TRUE., then it is assumed that f_{nn_0} is negligible at RSTART-H(1)/2. This is called the homogeneous option, since it implies that the coupled equations are solved subject to homogeneous boundary conditions ($f_{nn_0} \to 0$) at

the origin. In this case, just the R_4 submatrix of the global R matrix is propagated, and the program may both drop uncoupled channels from propagation {8.7} and from the final asymptotic analysis {8.12}. This option is preferred for heavy-particle collisions, since much of the work at small r becomes unnecessary. In addition, some of the time-saving options {8.10} are only allowable with this option.

8.6 If LR4MT = .FALSE., then the program will propagate the full R matrix at each sector. This options requires more memory (the storage of the full matrix and takes more time (propagation of the full R matrix takes at least one-third longer than propagation of just the R_4 submatrix, depending on the value of the option NPROP {8.10} used to propagate R_4). In particular, NPROP = 2, which gives a significant speedup for the homogeneous option, is invalid for the inhomogeneous option. In addition, the inhomogeneous option requires that the propagation be started closer to the origin. Finally, use of the inhomogeneous option means that no channels may be dropped from the propagation {8.7}, even though they may be removed prior to the asymptotic analysis. The use of the inhomogeneous boundary conditions is only recommended for physical systems where the collision partners may approach arbitrarily close to one another, such as electron-atom scattering. In such cases the radial functions do not become regular at the origin. For neutral heavy particle scattering, the inhomogeneous boundary conditions may still be used, but they are not necessary for an accurate solution (Section 3.5 of METECC-94 chapter and discussion following [87]).

8.7 At the center of each sector, we determine the matrix representation of scriptV in the primitive basis by a call to subroutine POT. This matrix is called D. It is assumed that the coupling between the radial functions is independent of r in each sector. We then can determine a local adiabatic basis by diagonalizing D to yield eigenvalues and eigenvectors. The matrix which diagonalizes D is used to form the overlap matrix [50], which is called TAUE in the program. Only the functions with the lowest PDIM eigenvalues are used in propagating across the sector. The initial value of PDIM is set to the input variable PD. When RABC(I) is greater than the input variable RBIG, then the program tries to reduce PDIM using the input tolerance EPSRED (discussed after [71] in the METECC-94 chapter). Only functions which would not be energetically allowed asymptotically are dropped from the propagation. It is also important to note that PDIM will remain constant (no channels dropped from propagation) if the inhomogeneous option {8.6} is used.

8.8 It should be noted that the eigenvectors of D are unchanged by adding a multiple of the unit matrix to D. Thus if another calculation is to be carried out which only changes D in this manner, the eigenvectors can be reused if they have been saved. In practice we save the overlap matrix rather than the eigenvectors. Such saving can be very advantageous because the computation of D and its diagonalization can be expensive. The program allows this information to be reused in four situations: (1) the total energy E changes, (2) the initial value PD of PDIM {8.7} changes, (3) the initial integration distance RSTART {8.4} changes, or (4) the value of the orbital angular momentum OAM for each channel changes. (OAM is used for spherically symmetric potentials where the orbital angular momentum is identical in all channels.) Currently E, RSTART, and OAM can all change with PD fixed, or PD can change with E, RSTART, and OAM

fixed. For succinctness, these are all called second-case runs {8.9}.

8.9 The program can perform second-case runs in two modes. In type-1 second-case runs, the propagation is completed for the first case before the second case is begun. This requires that the TAUE matrix {8.7}, and perhaps its inverse, which is called TINVE in the program, be saved on disk files at each sector {8.10}. This is done by setting the number of energies, NE, to a positive number, or by utilizing the IROWS options. In type-2 second-case runs, all cases are propagated across each sector together, and the TAUE and TINVE matrices need not be stored. This option is accessed by setting the number of energies to a negative number. For type-2 second-case runs, it is necessary to know at what energies the second and subsequent cases are to be run, while for type-1 second-case runs it is possible to save the disk files and perform second-case calculations at some later date. Type-2 second-case runs do not require disk storage, but do require additional memory since the R_4 matrix {8.5} or the full R matrix {8.6} for each case must be stored.

8.10 To perform the actual propagation from one sector to the next, several propagation formulas can be used. If NPROP is 0, [58e] is used exactly as it is written in the chapter. If NPROP is 1, then [58e] is used, except that the calculation of the sector r matrices $r_1 - r_4$ [53]-[56] which are used in [58e] is performed using the transpose of the TAUE {8.7} matrix rather than its inverse. If NPROP is 2, then [58e] is used, except that the local r matrices $(r_1 - r_4)$ are not stored explicitly. If PDIM {8.7} = NDIM {8.2}, in the absence of roundoff error, all formulas should give the same result. If PDIM is not equal to NDIM, then NPROP = 0 or 2 is more rigorous, although NPROP = 1 will always give a unitary scattering matrix while the other options will not. The NPROP = 2 formulas require less work than the other formulas. If type-1 second-case runs {8.9} are performed, NPROP = 1 and 2 do not require that the inverse of the TAUE matrix be stored on disk. If the inhomogeneous option {8.6} is used then NPROP = 2 is invalid, and [58a-e] are used throughout the propagation with explicit storage of the matrices $(r_1 - r_4)$. In general the usage of NPROP = 2 is the most efficient.

8.11 As the propagation proceeds, there can be many crossings of the eigenvalues of the D matrix {8.7}. When two or more eigenvalues are degenerate in the current sector, then the eigenvectors will be mixed. At intermediate distances, this is of no consequence, however, it poses a problem when carrying out the asymptotic analysis. This mixing can be resolved and the R matrix transformed from the local adiabatic (mixed) basis to the unmixed primitive basis [75]. This option is enabled by setting the input logical variable LTASY to .TRUE..

8.12 When carrying out the asymptotic analysis, it is possible to neglect closed channels. For the homogeneous option {8.5}, this is done by comparing the coupling between open and closed channels in the R_4 matrix {8.4} to the parameter EPSDR, as described after [86c]. For the inhomogeneous option {8.6}, this is done by comparing the coupling between open and closed channels in the M matrix [86b] as described after [86c].

8.13 When the asymptotic analysis is performed, several options are available for the matching functions. If LBES is true, Ricatti-Bessel functions are used;

otherwise sine/cosine functions are used. The wave vectors for each channel used in the matching functions can be determined in any one of three ways, specified by the variable IMATCH. If IMATCH is 0, then the wavevectors are determined from the eigenvalues of D {8.7} evaluated at r = RASYE. If IMATCH is 1, then the eigenvalues of D at r = RABC(i) {8.4}, where i is the current sector number, determine the wavevectors. Finally, if IMATCH is 2, then the eigenvalues of D at r = RABC(i) + H(i)/2 are used.

8.14 For situations where the reactance matrix is not symmetric because PDIM {8.7} was not large enough, it can be symmetrized by one of three options set by the input variables LGS(9) and IOPT3. The reactance matrix will only be symmetrized if LGS(9) is .TRUE.. If IOPT3 is 1, then the lower left triangle of the reactance matrix will be imposed upon the upper right. If IOPT3 is 2, then the upper right triangle of the reactance matrix will be imposed on the lower left. If IOPT3 is 3 then the reactance matrix will be symmetrized by using the arithmetic mean of the original pair of off-diagonal elements. IF IOPT3 is 4 then each of the methods will be employed in turn, but the scattering matrix will be calculated from the reactance matrix symmetrized using the arithmetic mean of the off-diagonal elements.

8.15 The asymptotic analysis will be performed either at the intermediate distances defined by RPSM, when the maximum propagation distance RMAX is exceeded, or when the maximum number of sectors, NSECT is exceeded. If possible the program attempts to stop the calculation before RMAX is reached. This is done by comparing the results of the current asymptotic analysis to the one performed previously. For type-2 second-case runs, this is done only for the first case. If the difference is small enough, the calculation stops. If EPSMAG is nonzero, then the NIND matrix elements of the scattering matrix specified by the array IND are compared, and if the relative differences in the magnitudes of all NIND matrix elements are less than EPSMAG, then convergence is assumed. If in addition, EPSPH is also nonzero, then the relative differences in the phases of the NIND matrix elements are also compared to EPSPH, and convergence is assumed only if both the EPSMAG and EPSPH tests are passed. If both EPSMAG and EPSPH are zero, the calculations are not stopped until RMAX is reached. In addition, the program contains a trap which prevents the asymptotic analysis from being performed at a given RPSM if the asymptotic analysis for the previous RPSM was performed in the previous sector; the program will not perform the asymptotic analysis for the RPSM for two adjacent sectors. This is to prevent the user from assuming convergence with respect to increasing the propagation coordinate by comparing the asymptotic analysis at too closely spaced distances.

9. UNITS

The code is set up to be valid for any consistent set of units for which the Hamiltonian has the form of the equations in {8.2}. This is most conveniently accomplished by using Hartree atomic units, given below. All test runs utilize these units.

length	bohr	5.2918E-10 m
energy	hartree	4.3592E-18 J
angular momentum	h/(2 * pi)	1.0546E-34 J*s
mass	electron mass	9.1095E-31 kg

10. CALCULATIONS WITH A MODULAR POTENTIAL MATRIX ELEMENT ROUTINE

This section discusses the subroutines required by RMPROP to integrate the close coupling equations when the user supplies a FORTRAN source code to generate the scriptV(r, x) matrix. It includes a listing of the files, argument lists, and a description of each argument. Numbers in {} refer to paragraphs in Section 8 above.

The program is distributed with potential function subprograms for the test runs, but most users will need to supply other potential function subprograms corresponding to their own problems. This section of the manual describes specifications necessary for the user to run this program with his or her own potential matrix element subprogram. The subprogram is required to supply to RMPROP a matrix which contains in location (I,J) the interaction potential integrated between asymptotic basis functions I and J at the current value of the radial separation, described in {8.7}. For certain forms of the interaction potential and collision partners, these integrals are known analytically, or they may be evaluated numerically by specific quadrature methods. If the interaction potential is already in such a form that the integrals are known analytically, then the modular potential matrix element routine should be used to supply the interaction potential matrix elements to the main program. (The case of a general interaction potential is discussed below, in Section 11 of this manual.)

Examples of "standalone" potential matrix element generation routines are in Test Suite elements hfhfstr1-5, heh2cdr1-2, hfhfpbsr1, and ehydr1 (see Section 19).

USER REQUIRED:

SUBROUTINE PREPOT	See Sections 10.1 and 11.
SUBROUTINE POT	See Sections 10.1 and 11.
parameter.inc	See Sections 10.2, 11.3.2, 14, and 15.
fort.5 (Main input.)	See Section 13.

10.1 DESCRIPTIONS OF SUBPROGRAMS

The call to PREPOT is

```
CALL PREPOT(LSTAP,NDIM,JIN,ID,L,JTOT,B,IS,EI,EFASY)
```

PREPOT is designed to set variables, read information, and write messages pertaining to the potential. These are tasks which it is assumed need only be performed once for any particular potential. Messages should be written to units 6 (long output) and 15 (short output). Information may be read from unit 4, which has been reserved for potential-defining input.

The call to PREPOT occurs after FORTRAN unit 5 has been read. In particular PREPOT is called once if only one scattering potential is used and once for each potential if more than one is used. The only input information read in for second and subsequent potentials is that read in by subroutines PREPOT (to define the potential) and by PRSCAT (to govern the printing of scattering output for that potential).

PREPOT must set the array EI, and if LTASY is true, it must also set the array EFASY. The arrays B, JIN, and IS are used only by PREPOT and POT, typically to store rotational and vibrational quantum numbers, respectively. For correct dimensioning, PREPOT must also incorporate the FORTRAN statements

```
INCLUDE 'parameter.inc'
DIMENSION EI(IDX),B(IDX,IDX,NVIB1X),JIN(IDX,NANX),IS(IDX,NVIBX)
DIMENSION L(IDX),EFASY(IDX,IDX)
```

Note: some options require that the channels are ordered by energy (e. g. LTASY {8.11}). To facilitate this it is permissible for PREPOT to reorder the elements of L, JIN, IS, and EI.

Calling Parameters:

LSTAP: a logical variable. By default, RMPROP sets LSTAP to .TRUE., so that RMPROP will stop after the propagation is complete for that potential. If LSTAP is reset to .FALSE. in PREPOT, PREPOT will be called again to define a new potential, and another scattering calculation will begin. RMPROP will continue to complete propagations and read in new potentials from PREPOT until LSTAP is set to .FALSE. at the beginning of the final scattering calculation. The records read in again for the new scattering calculations are records 33 and following from FORTRAN unit 5, read in by subroutine PRSCAT, and any records required by the potential subroutine, read in from FOTRAN unit 4.

NDIM: an integer variable set by the main program (read in from FORTRAN unit 5); it is the number of primitive basis functions in the calculation {8.2}.

JIN: an integer array dimensioned JIN(IDX,NANX), set by PREPOT. IDX and NANX are parameters set in the include file parameter.inc. Only PREPOT and POT use the array, which is typically used to store angular momentum quantum numbers other than the orbital angular momentum for relative translational motion. In test suite elements, JIN stores the rotational quantum numbers of diatoms for each channel and the coupling of rotational quantum numbers for the diatoms in each channel.

ID: an integer variable set by RMPROP (read in from FORTRAN unit 5). It is usually equal to IDX in the file parameter.inc, and is no longer used.

L: an integer array dimensioned L(IDX) set by RMPROP when read in from FORTRAN unit 5. It is the quantum number specifying the orbital angular momentum quantum number for relative translational motion {8.2}. The elements of L may be reordered by PREPOT if necessary (e.g., channels are sorted by energy in PREPOT).

JTOT: an integer variable set by RMPROP after being read in from FORTRAN unit 5. It is the total angular momentum quantum number {8.2}.

B: a double precision array dimensioned (IDX,IDX,NVIB1X). This array is never used by the main program and may be used to supply scratch space for PREPOT and POT.

IS: an integer array dimensioned IS(IDX,NVIBX), set by PREPOT. Normally, the vibrational quantum numbers of the collision partners are stored here.

EI: a double precision array of dimension EI(IDX) set by PREPOT which must contain the asymptotic channel eigenenergies in atomic units for the collision partners.

EFASY: a double precision array dimensioned EFASY(IDX,IDX) and set by PREPOT. EFASY is required only if LTASY (read in from FORTRAN unit 5) is .TRUE. and in that case should contain the eigenvectors of the asymptotic basis in the primitive basis {8.11}. Normally this would be the unit matrix.

The call to POT is

```
CALL POT(RABC,NDIM,ID,B,UPCDMT)
```

POT is expected to calculate the potential matrix UPCDMT. POT should incorporate the FORTRAN statements

```
INCLUDE 'parameter.inc'
DIMENSION B(IDX,IDX,NVIB1X),UPCDMT(IDX,IDX)
```

CALLING PARAMETERS:

RABC: a double precision variable set by RMPROP. This is the value of the scattering coordinate r for which the potential matrix is to be calculated.

NDIM: see PREPOT above.

ID: see PREPOT above.

B: see PREPOT above.

UPCDMT: a double precision array of dimension UPCDMT(IDX,IDX) calculated by POT. UPCDMT contains the matrix elements of the potential scriptV(r, x) {8.2, 8.7} in the primitive basis, evaluated at r.

10.2 INCLUDE file

The file parameter.inc must be present in the source code directory when the code is compiled. The contents of the file should be the fortran source code statement:

```
      PARAMETER(IDX=ii1,NSECTX=ii2,NANX=ii3,NSAVEX=ii4,NVIBX=ii5,NVIB1X=ii6,
     & NNEX=ii7,NBARX=(IDX*(IDX+1)/2))
```

where ii1, ii2, ... represent integers which are constant within each run but which may vary from run to run. These values are used to dimension many of the arrays within the program. The file parameter.inc and the parameter list are discussed in more detail in Sections 14 and 15.

11. CALCULATIONS WITH A USER-SUPPLIED POTENTIAL ENERGY SURFACE SUBPROGRAM

11.1 DESCRIPTION OF GENERAL POTENTIAL MATRIX ROUTINES

In RMPROP-Version 2.1.2 it is possible (for some types of systems) for the user to supply a general form for the interaction potential between the collision partners and to allow RMPROP to numerically evaluate the matrix elements of the interaction potential between asymptotic basis functions {8.2, 8.7} which it needs at each sector of the propagation.

The user can supply one of three types of potential surfaces, and RMPROP will form the scriptV(r, x) matrix from the surface, as follows:

1) Atom-rigid rotor surface, with potential expanded in Legendre functions times radial functions. This will be discussed in more detail in Section 11.2.

2) General Atom-(vibrating) rotor surface. This will be discussed in more detail in Section 11.3.

3) Diatom-diatom surface, when diatoms are identical. This will be discussed in more detail in Section 11.4.

The numerical evaluation of the integrals is carried out in several steps which are described below.

The first step is the expansion of the user-supplied potential in a set of convenient angular functions. The type of angular functions used will depend upon the system being studied. For an atom-diatom system there are two options. If the diatom is being treated as a rigid rotor, and the interaction potential is already expressed solely as a summation of Legendre functions times radially-dependent coefficients, then vgr1.f should be used. See Section 11.2 (the discussion of vgr1.f) and Sections 19.5 and 19.11 (test runs hehfr1 and atdir1) for examples of this. If the diatom is a vibrating rotor or the atom-diatom potential is not expanded in Legendre functions, then vgr2.f should be used. See Section 11.3 (the discussion of vgr2.f) and Sections 19.13-15, (test runs hei2r1, heh2tkr1, and heh2tkr2) for examples of the vibrating rotor, and Section 19.16 (test run heh2tkr3) for an atom-rigid rotor calculation. For a diatom-diatom system where the two diatoms are identical, the interaction potential is expanded in Launay functions, using vgr3.f. Depending upon the input options, either rigid rotors or vibrating rotors may be treated. See Section 11.4 (for a discussion of vgr3.f) and Sections 19.9 and 19.10 (test runs hfhfadr1 and hfhfadr2) for examples of the use of vgr3.f with a rigid rotor).

The second step is the integration of the angular functions used in the expansion of the interaction potential with the angular-dependent factors of the appropriate asymptotic channel basis functions. The choice of the angular functions used to expand the interaction potential allows these integrals to be evaluated analytically. (These angular integrals may be saved to disk on FORTRAN unit 2 if desired; however it should be noted that this requires a large amount of storage).

The third step is the evaluation (if necessary) of the vibrational quadratures [A.28] contributing to the potential matrix elements at each sector. This is performed in two steps. The vibrational contribution to the potential matrix elements are found by Gaussian quadrature in prevgr2.f or prevgr3.f, and so the weights and nodes for the quadrature must first be determined. This choice of weights and nodes is performed in SUBROUTINE WEIGH, discussed in Section 11.3.3 below. First, weights and eigenvalues are found for the integration of the vibrational wave function for the diatom in the system under study. This is done by expanding the wave functions in terms of a harmonic oscillator basis set, and doing all integrals required in the harmonic basis. The nodes may be found in one of four ways: semi-classical nodes, scaled Gauss-Hermite nodes, Gaussian nodes for the weight (which is the square of the vibrational wave function), or calling a user-supplied routine SDATA to supply the nodes. After the determination of the weights and nodes, the Gaussian quadrature is performed at each step of the propagation to determine the vibrational contribution to the interaction potential matrix (the scriptV (r, x) matrix of {8.2, 8.7}).

The last step is the combination of the angular and the vibrational contributions [A.26] to the interaction potential matrix at each sector of the propagation to supply the overall interaction matrix. There are two additional features of the expansion/quadrature of an arbitrary interaction potential which should be noted. First, the program allows the number of angular terms used in the angular expansion of the potential, and the order of quadrature of the vibrational contribution to the interaction potential, to be decreased as a function of the radial separation. This is useful for systems where the anisotropy of the interaction potential decreases strongly with increasing radial separation, since then a fewer number of terms may be needed for an accurate description of the interaction potential. Second, the order of the angular expansion and of the vibrational quadrature may be varied independently at the discretion of the user.

It should also be pointed out that both the expansion of the interaction potential in a set of known angular functions and the quadrature over the vibrational contribution to the interaction potential matrix are numerical. Therefore, any scattering matrix elements elements obtained from the run should be checked for convergence both with respect to the number of known angular functions in which the interaction potential has been expanded and with respect to the angular quadratures which determined the expansion coefficients of the interaction potential.

11.2 POTENTIAL MATRIX ELEMENT ROUTINE vgr1.f

This section of the manual describes the usage of the subroutine vgr1.f, which is intended for use with 3-dimensional atom-rigid diatom systems where the interaction potential is already explicitly written in terms of Legendre functions. Subroutine vgr1.f differs from both vgr2.f and vgr3.f in that vgr1.f can ONLY be used with rigid rotors, so that there is no provision for the vibrational contribution to the scriptV (r, x) matrix. The test suite runs hehfr1 and atdir1 are examples of the use of vgr1.f.

The user-supplied form of the interaction potential for vgr1.f is unique in that vgr1.f assumes that the interaction potential is already written in terms of Legendre functions times radial-dependent coefficients. Therefore, any atom-diatom interaction potential in which the interaction is not expressed as a Legendre expansion should be used with vgr2.f (see Section 11.3 below).

Examples of the use of prevgr1.f and vgr1.f are in the Test Suite elements hehfr1 and atdir1, with the code in pothehf.f and potatdi.f, respectively.

USER REQUIRED:

SUBROUTINE HEADER	see below
SUBROUTINE FUN1	see below
DOUBLE PRECISION FUNCTION EIGEN	see below
parameter.inc	see Section 10.2, Sections 14, 15
fort.4	see below
fort.5 (main input)	see Section 13

11.2.1 List of User-supplied subroutines for use with vgr1.f

The calling sequence for SUBROUTINE HEADER is

```
SUBROUTINE HEADER
```

The body of this subroutine should include all write statements concerning the potential used in the calculation, with the appropriate units being unit 6 for the long output and unit 15 for any short output.

The calling sequence for the user-supplied potential energy surface is

```
SUBROUTINE FUN1(R,LMAXP1,VL)
```

R: Double precision variable, input. This is the propagation coordinate in bohrs.

LMAXP1: Integer variable, input. This is the maximum order of legendre function in the potential expansion plus one; LMAXP1= LAMMAX + 1, where LAMMAX is described in Section 11.2.2 above.

VL: Double precision vector, length LMAXP1, output. Upon return, this vector should hold the radial-dependent coefficients of the Legendre functions used to describe the interaction potential. For the case of a homonuclear diatomic, only even values of legendre functions will have nonzero coefficients. Subroutines vgr1.f and prevgr1.f expect that for this case, VL will hold only these nonzero coefficients in the first LAMMAX/2 +1 positions of the vector.

The calling sequence for function EIGEN is
EIGEN(IV,J)

IV: Input, integer dummy variable. This variable indicates the vibrational quantum number when this function operates for vibrating rotors (as in conjunction with vgr2.f, vgr3.f). This variable is set to zero by PREPOT in prevgr1.f and should have no affect on the value of EIGEN.

J: Input, integer variable. This is the rotational quantum number of the diatom for the channel of interest.

EIGEN: Output, double precision. The function should assign to EIGEN the energy in hartrees of the asymptotic channel with the diatom in rotational state J.

11.2.2 INPUT TO FORTRAN UNIT 4

CARD	Variable name	Format
1	LSTAP	L1
2	JIN(I,1)	10I5
3	IHOM,LAMMAX	10I5

LSTAP is a logical variable ordinarily set equal to .TRUE.. It is used to tell RMPROP whether to do more than one approximation to the interaction potential during a run, i. e., if LSTAP .EQ. .TRUE., then the current potential is the only one used.

(JIN(I,1),(I=1,NDIM)) contains the internal rotational angular momentum quantum numbers for the diatom.

IHOM should equal 2 for homonuclear and 1 for heteronuclear diatoms.

LAMMAX is the maximum index of the Legendre functions in the potential.

11.3 POTENTIAL MATRIX ELEMENT ROUTINE vgr2.f

For users with an atom-diatom potential energy surfaces but little interest in writing matrix-element generating routines, vgr2.f and prevgr2.f offer a modular interface to RMPROP 2.1.2.

This section describes the usage of the subroutines in vgr2.f, which is intended for use with atom-vibrating diatom systems in 3 dimensions. The subroutines in vgr2.f allow for vibrational contributions to the scriptV (r, x) matrix {8.2,8.7}, by calls to SUBROUTINE WEIGH and SUBROUTINE WDATA, which calculate weights and nodes to solve the vibrational integrals by quadrature. It is also possible to use prevgr2.f and vgr2.f for rigid rotor systems (see variable NSAVE on RECORD 14 of input to UNIT 4 below).

The test suite elements which employ prevgr2.f and vgr2.f are hei2r1, heh2tkr1, and heh2tkr2, with source codes in pothei2.f and tkrpot.f.

USER REQUIRED:

SUBROUTINE HEADER	see 11.2.1
SUBROUTINE FUN1	see below
SUBROUTINE PTNTL	see below
DOUBLE PRECISION FUNCTION EIGEN	see below
parameter.inc	see below, 14, 15
expand.inc	see below
fort.4	see below
fort.5	see 13

11.3.1 USER-SUPPLIED SUBROUTINES FOR vgr2.f

The call to HEADER was described in the previous section.

The calling sequence for subroutine FUN1, which returns the value of the interaction potential as a function of atom, diatom coordinates, is

```
SUBROUTINE FUN1(R,R1,COSC,FN,NPTS)
```

R: Double precision variable, input. This is the value of the propagation coordinate of the atom and diatom in bohr.

R1: Double precision variable, input. This is the bond length of the diatom in bohr.

COSC: Double precision vector of length NPTS, input. This vector contains the cosine of the orientation angle chi at the angular quadrature points. Chi is the angle between R and R1.

FN: Double precision vector of length NPTS, output. FN(I) is the interaction potential in hartrees evaluated at the coordinates R, R1, COSC(I).

NPTS: Integer variable, input. This is the number of points at which the potential should be evaluated. (It is equal to the current order of the angular quadrature.)

The calling sequence for the (user-supplied) diatomic potential curve is

```
PTNTL(X)
```

X: Double precision variable, input. X is the value of R - RE, where R is the bond length of the diatom in bohr.

PTNTL: Double precision, output. The function should assign to PTNTL the energy of the diatomic potential at X in hartrees, and should return a value of 0 for X = 0.

The calling sequence for the function EIGEN is

```
EIGEN(IV,J)
```

IV: Integer variable, input. The vibrational quantum number of the diatom for the channel of interest.

J: Integer variable, input. The rotational quantum number of the diatom for the channel of interest.

EIGEN: Double precision, output. The asymptotic channel eigenenergy in hartrees.

Note: The user may utilize the asymptotic eigenergies of diatom v,j states from SUBROUTINE WEIGH and WDATA by using the input variable IEIGN of RECORD 11 on UNIT 4 (see below). This is not permitted for rigid rotors or for vibrational states calculated without rotational coupling (IVIBJ.NE.0, JSAVE.EQ.0 on UNIT 4).

Finally it should be noted that if the value of INODE read in from FORTRAN unit 4 (see Section 11.3.3 above) is 1, 2, or 3, the user should supply a dummy SUBROUTINE SDATA(NPTS,XS) to avoid warning messages on loading.

11.3.2 INCLUDE FILES

The user must supply the file parameter.inc in the source code directory during compilation of the code. The contents of this file must be the fortran statement

```
      PARAMETER(IDX=ii1,NSECTX=ii2,NANX=ii3,NSAVEX=ii4,NVIBX=ii5,NVIB1X=ii6,
     & NNEX=ii7,NBARX=(IDX*(IDX+1)/2))
```

where ii1, ii2, ... represent integers which are constant within each run but which may vary from run to run. These values are used to dimension many of the arrays within the program. The variables within parameter.inc are described in more detail in Sections 14 and 15 below, but we note here that NANX should be at least 1, NVIBX should be at least 1, and NVIB1X should be set large enough that the array BPOT(IDX,IDX,NVIB1X) is large enough to hold the angular integrals required by the program. The number of words needed by BPOT may vary considerably, since only nonzero values of the integrals are stored. An upper limit on the value of NVIB1X is 3*NUMT, where NUMT is the number of terms in the angular potential expansion (see RECORD 16 input to FORTRAN UNIT 4 below).

The exact number required within a given run is written to standard output with the legend

... working precision words of bpot used in pot

where ... is an integer representing how many words of storage were required by the angular integrals. If the angular integrals would have required too much space, the program instead prints the message

Dimensions exceeded in angint. IBMX= ...

where ... is the current value of IDX*IDX*NVIB1X.

The second include file, expand.inc, should contain the fortran statement

PARAMETER (NQAMAX=...,NQVMAX=...,MAXV=...,MAXJJ=...,NUMBT=...)

where the ... indicates an integer, and where the terms have the following meanings:

NQAMAX must be greater than or equal to the maximum order of angular quadrature. It is used to dimension arrays concerned with the Legendre expansion.

NQVMAX must be greater than or equal to the maximum order of vibrational quadrature. It is used to dimension arrays concerned with the vibrational quadrature.

MAXV must be greater than or equal to the maximum vibrational quantum number + 1. It is used to dimension the array of eigenvalues returned from SUBROUTINE WEIGH.

MAXJJ must be greater than or equal to the maximum rotational quantum number +1 of the diatom. It is used to dimension the array containing the weights for the vibrational quadrature.

NUMBT must be greater than or equal to the maximum number of Legendre functions used in the potential expansion (NUMT input in UNIT 4).
It is used to dimension several arrays related to the expansion of the interaction potential in Legendre functions.

11.3.3 SUMMARY OF INPUT TO FORTRAN UNIT 4

A record-by-record summary of the input to the RMPROP code from FORTRAN unit 4 when module vgr2.f and its associated subroutines are used follows. Note that the number of fields within each record may not match the FORMAT given because some of the FORMAT statements are used for other purposes with the subprograms.

Record number	Input variables	Format
1	RE	9E15.8
2	NUMQ	10I5
3	(NQUAD(I),I=1,NUMQ)	10I5
4	(NQUADA(I),I=1,NUMQ)	10I5
5*	(RQUAD(I),I=1,NUMQ-1)	9E15.8
6	LSTAP	L1
7	NLINE	10I5
8	MES	80A8
9	(ISJ(I,1),I=1,NDIM)	10I5
10	(JIN(I,1),I=1,NDIM)	10I5
11	IHOM,IVIBJ,IEIGN	10I5
12	RMS,OMEGA,RCENT,OMEGGH,LRCENT,SCAL2	4E15.7,L1,E15.7
13	NHQD,NBASIS,NPTSM,IPRINT,INODE,IWGHT	6I5
14	NINT,NL,NSAVE,JSAVE,IROTN	5I5
15%	NPTS	6I5
16@	NUMT	I5

*Only if NUMQ.GT.1
%Only if NSAVE.NE.0; Then card repeated until a value of zero given for NPTS
@Card repeated NUMQ times

11.3.4 EXPLANATION OF INPUT VARIABLES TO FORTRAN UNIT 4

Record 1: RE: (9E15.8)

This variable is the equilibrium bond length of the diatom in bohr.

Record 2: NUMQ: (10I5)

This variable is the number of different orders of quadrature (vibrational and/or angular) to be used during the propagation.

Record 3: (NQUAD(I),I=1,NUMQ): (10I5)

This variable is the number of points to be used in the vibrational quadrature for R less than RQUAD(I), where RQUAD (I) is discussed in Record 5 below. If one wishes to use a rigid rotor, NQUAD is set to 1, and the flag governing the use of a rigid rotor is set by NSAVE in Record 14 (see below).

Record 4: (NQUADA(I),I=1,NUMQ): (10I5)

This variable is the number of points to be used in the angular quadrature for R less than RQUAD(I), where RQUAD(I) is discussed in Record 5 below. Note that the separate specification of NQUAD and NQUADA allows the orders of the two quadratures to be varied independently.

Record 5: (RQUAD(I),I=1,NUMQ-1): (9E15.8) (Only if NUMQ > 1)

This variable governs the value of the propagation coordinate R such that the order of the vibrational and/or rotational quadrature will change for R .GT. RQUAD(I). See also Section 23. Only given if NUMQ > 1.

Record 6: LSTAP: (L1)

This variable governs whether just one interaction potential is used. It is a logical variable ordinarily set equal to .TRUE.. It is used to tell RMPROP whether to do more than one approximation to the interaction potential during a run, i. e., if LSTAP .EQ. .TRUE., then the current potential is the only one used. If LSTAP .EQ. .FALSE., then following the end of the first run, RMPROP will cycle back and attempt to read parameters for a new potential from FORTRAN unit 4.

Record 7: NLINE: (10I5)

This variable is a control variable used to define the number of lines of text to be read from FORTRAN unit 4 in Record 8.

Record 8: MES: (A80)

This variable consists of NLINE lines of text. These lines of text are ordinarily used to record details about the individual test run or the potential surafce being used.

Record 9: (ISJ(I,1),I=1,NDIM)): (10I5)

This variable contains the internal vibrational quantum numbers of the diatom.

Record 10: (JIN(I,1),I=1,NDIM)): (10I5)

This variable contains the internal rotation angular momentum quantum number of the diatom.

Record 11: IHOM,IVIBJ,IEIGN: (10I5)

These variables have the following meanings:

IHOM is a switch governing whether the diatom is heteronuclear(=1) or homonuclear(=2), for the purposes of calculating the indices of the Legendre functions used to expand the potential. If IHOM=1, IROTN (RECORD 14) must be zero. If IHOM=2, IROTN must be either 1 or 2, and NUMT (RECORD 16) represents the number of even terms in the Legendre expansion.

IVIBJ is a switch which determines whether rotational coupling is included in the calculation of of the vibrational quadrature.
IVIBJ .EQ. 0 means use j-dependent vibrational states (rotational coupling).
IVIBJ .NE. 0 means use j-independent vibrational states (no rotational coupling). For this case, JSAVE (RECORD 14) must be zero.

IEIGN governs whether or not the eigenenergies are calculated during the calculation of the vibrational weights and nodes or are calculated using the (user-supplied) FUNCTION EIGEN.
IEIGN .EQ. 0 means get channel eigenenergies from EIGEN.
IEIGN .NE. 0 means get channel eigenenergies from the vibrational weights and nodes computation (NOT POSSIBLE FOR RIGID ROTORS OR IF IVIBJ .NE. 0).

Record 12: RMS,OMEGA,RCENT,OMEGGH,LRCENT,SCAL2: (4E15.7,L1)

These variables are used by SUBROUTINE WEIGH which calculates the weights and nodes for the vibrational quadratures which compute the vibrational portion of the interaction potential matrix. This routine calculates the weights for the quadrature given the (user-supplied) diatomic potential curve (e. g. a harmonic oscillator or a Morse), by transforming the wave function into a harmonic oscillator basis and performing the integrals in that basis.

RMS is the reduced mass of the diatom in atomic units.

OMEGA is the angular frequency of the harmonic oscillator used in the basis.

RCENT is the equilibrium bond length of the oscillator. The (user supplied) diatomic potential function.

OMEGGH is the frequency used to find scaled Gauss-Hermite nodes. If OMEGGH is zero, OMEGA is used instead.

LRCENT is a logical variable.
If LRCENT .EQ. .TRUE. calculate the value of the wave function at -RCENT.

Record 13: NHQD,NBASIS,NPTSM,IPRINT,INODE,IWGHT: (6I5)

These variables are used by SUBROUTINE WEIGH which calculates the weights and nodes for the vibrational quadratures which compute the vibrational portion of the interaction potential matrix.

NHQD is the order of the Gauss-Hermite quadrature used to evaluate matrix elements in the harmonic oscillator basis.

NBASIS is the number of harmonic oscillator states to include in the expansion.

NPTSM is the maximum size of the vibrational quadrature to be found. This must be greater than or equal to all values of NQUAD (RECORD 3).

IPRINT is an inputinteger control variable. Set IPRINT .GE. 0 for various debugging output to standard output. This should be used cautiously because it can generate large quantities of output.

INODE determines the type of nodes used:

```
INODE= 1 use semi-classical nodes.
INODE= 2 use scaled Gauss-Hermite nodes.
INODE= 3 use Gaussian nodes for the weight function PSI(IWGHT)**2.
INODE= 4 call SDATA to get other (user-supplied) nodes.
```

It should be noted in passing that although SUBROUTINE WEIGH has been tested with all values of INODE, the elements of the test suite included in RMPROP-Version 2.1.2 only include the values 1 and 3. In addition, the user should supply a dummy version of SDATA to prevent error messages on loading.

IWGHT runs from 1 upwards, and is used in conjunction with INODE .EQ.3.

Record 14: NINT,NL,NSAVE,JSAVE,IROTN: (4I5)

These variables are used by SUBROUTINE WEIGH which calculates the weights and nodes for the vibrational quadratures which compute the vibrational portion of the interaction potential matrix.

NINT and NL only have meaning when INODE .EQ. 1 (semiclassical nodes).

NINT is the number of integration intervals to be used by SUBROUTINE NODE, which calculates semi-classical nodes.

NL is the order of Legendre interpolation used by SUBROUTINE NODE, which calculates semiclassical nodes.

NSAVE is the maximum vibrational quantum number + 1 for which weights and nodes are to be saved on FORTRAN unit 1. If NSAVE is 0, special output is generated for a rigid rotor. In addition, then, for NSAVE=0, NQUAD (RECORD 3) must be 1, JSAVE must be zero, IVIBJ (RECORD 11) must not be zero, and IEIGN (RECORD 11) must be zero. (The User must supply DOUBLE PRECISION FUNCTION EIGEN.) Also, for NSAVE=0, RECORD 15 cannot exist.

JSAVE is the maximum rotational quantum number + 1 of rotational states in the calculation of the vibrational states. If no rotational-vibrational coupling is desired, use JSAVE=0. IVIBJ cannot be zero for JSAVE=0.

IROTN is zero for heteronuclear diatoms (or homonuclear with all J states), one for homon
+ uclear with odd J states, and two for homonuclear diatoms with even J states. When IROTN=1 or 2, IHOM (RECORD 11) must be 2.

Record 15: NPTS: (6I5) (ONLY if NSAVE.NE.0, see RECORD 14)

This card is repeated until a value of zero is given for NPTS. Each value of NPTS given is a vibrational quadrature order which will be used by SUBROUTINE WEIGH to generate weights and nodes written to UNIT 1. Each value of quadrature order in NQUAD must be repeated here. If NSAVE=0, this card cannot exist.

Record 16: NUMT: (10I5) (NUMQ times)

NUMT is the number of terms in the Legendre expansion for the potential. If IHOM (RECORD 11) is 2, NUMT should be the number of even terms in the expansion of the potential (odd terms do not contribute for a homonuclear diatomic). This variable is read in NUMQ times by SUBROUTINE ANGINT in vgr2.f, first when called by PREPOT in prevgr2.f, and NUMQ-1 times when called by POT in vgr2.f, whenever a value of RQUAD(I) (see Record 5 above) is passed during the propagation. NUMT is then the new number of terms in the Legendre expansion for values of the propagation coordinate between RQUAD(I) and RQUAD(I+1) (for I less than NUMQ -1), and for values of the propagation coordinate greater than RQUAD(I) (for I = NUMQ). Note that the angular integral matrix discussed in Section 11.3.1 above will be re-written to allow for the new value of NUMT.

11.4 POTENTIAL MATRIX ELEMENT ROUTINE vgr3.f

This section describes the usage of the routines in prevgr3.f and vgr3.f, which provide a modular interface to RMPROP 2.1.2 when a User has an interaction potential surface for a diatom-diatom system where the diatoms are identical. These routines may be employed for vibrating rotors or rigid rotors (see Section 19.?, test runs hfhfadr1 and hfhfadr2). At this time only identical diatoms can be used since the functions EIGEN and PTNTL are the same for each.

USER REQUIRED:

SUBROUTINE FUN1	see below
DOUBLE PRECISION FUNCTION EIGEN	see 11.3.1
DOUBLE PRECISION FUNCTION PTNTL	see 11.3.1
parameter.inc	see below, 14, 15
expand.inc	see below
fort.4	see below
fort.5 (Main input)	see 13

11.4.1 USER-SUPPLIED SUBPROGRAMS

This section gives the required calling sequence for user-supplied potential energy subroutines required in conjunction with the use of vgr3.f. These routines return the interaction potential between the collision partners, and the diatomic potential curve for the diatom. The user-supplied potential energy subroutine is required to return the value of the interaction potential for the collision partners in Hartree atomic units given a specific choice of the coordinate system, in order to expand the interaction potential in Launay functions. The diatomic potential curve is required to calculate quadrature nodes used to deterimine the vibrational contribution to the interaction potential matrix at each sector.

The calling sequence for the user-supplied potential energy surface is

```
SUBROUTINE FUN1(R,R1,R2,C1,S1,C2,S2,CP,SP,V,NPTS)
```

R: double precision, input. The value of the propagation coordinate of the diatoms in bohr.

R1: double precision, input. The bond length of the first diatom in bohr.

R2: double precision, input. The bond length of the second diatom in bohr.

C1: double precision array of length NPTS, input. Used to hold the NPTS cosines of the angular quadrature nodes in angle THETA1.

S1: double precision array of length NPTS, input. Used to hold the NPTS sines of the angular quadrature nodes in angle THETA1.

C2: double precision array of length NPTS, input. Used to hold the NPTS cosines of the angular quadrature nodes in angle THETA2.

S2: double precision array of length NPTS, input. Used to hold the NPTS sines of the angular quadrature nodes in angle THETA2.

CP: double precision array of length NPTS, input. Used to hold the NPTS cosines of the angular quadrature nodes in angle PHI (= PHI2 - PHI1).

SP: double precision array of length NPTS, input. Used to hold the NPTS sines of the angular quadrature nodes in angle PHI (= PHI2 - PHI1).

V: double precision array of length NPTS, output. V(I) should contain the value of the interaction potential at the geometries specified by R1, R2, THETA1(I), THETA2(I), PHI(I). Note that THETA1, THETA2, and PHI are only implicitly specified by the arrays C1, S1, C2, S2, CP, and SP. There should only be NPTS nonzero results in V upon return.

NPTS: integer, input. This is the current number of angular quadrature points at which to determine the potential.

11.4.2 INCLUDE FILES

The user must supply the file parameter.inc in the source code directory during compilation of the code. The contents of this file must be the fortran statement

```
      PARAMETER(IDX=ii1,NSECTX=ii2,NANX=ii3,NSAVEX=ii4,NVIBX=ii5,NVIB1X=ii6,
     & NNEX=ii7,NBARX=(IDX*(IDX+1)/2))
```

where ii1, ii2, ... represent integers which are constant within each run but which may vary from run to run. These values are used to dimension many of the arrays within the program. The variables within parameter.inc are described in more detail in Sections 14 and 15 below, but we note here that NANX should be set equal to at least 3, NVIBX should be set to at least 2, and NVIB1X should be set large enough that the array BPOT(IDX,IDX,NVIB1X) is large enough to hold the angular integrals required by the program. (Both the calculation and the storage of the angular integrals in vgr3.f differ from that in vgr2.f). The number of words needed by BPOT may vary considerably since only nonzero integrals are stored, but an upper limit for NVIB1X should be 3*NUMT, where NUMT is the maximum number of terms in the Launay expansion of the potential. The exact number required within a given run is written to standard output with the legend

```
      ....   working precision words of BPOT used in POT
```

where .... is an integer representing how many words of storage were required by the angular integrals.

If the angular integrals would have required too much space, the program instead prints the message

```
                  Dimensions exceeded in angint. IBMX= ...
```

where ... is the current value of IDX*IDX*NVIB1X, and stops with the message

```
            STOP Try making NVIB1X larger in parameter.inc
```

which is the appropriate remedy.

The second include file should contain the statement

```
      PARAMETER(NUMBT=IA, NQAMAX=IB, NQVMAX=IC, MAXJJ=ID, MAXV=IE)
```

where the IA-E indicates an integer, and the terms have the following meanings:

NUMBT is greater than or equal to the maximum number of terms in the expansion of the interaction potential in Launay functions. It is used to dimension arrays which hold the indicies of the Launay functions, and to dimension an array which holds the expansion coefficients at each point in the vibrational quadrature for each term in the Launay expansion.

NQAMAX is greater than or equal to the maximum order of the angular quadrature. It is used to dimension arrays containing the sines and cosines of the nodes for the angular quadrature.

NQVMAX is greater than or equal to the maximum order of the vibrational quadrature. It is used to dimension arrays containing the nodes of the vibrational quadrature, the weights of the vibrational quadrature, and arrays relating the channel index to the position of the weight for that channel.

MAXJJ must be greater than or equal to the maximum rotational quantum number+1. It is used to dimension the array containing the weights for the vibrational quadrature.

MAXV must be greater than or equal to the maximum vibrational quantum number+1.

11.4.3 SUMMARY OF INPUT TO FORTRAN UNIT 4

A record-by-record summary of the input to the RMPROP code from FORTRAN unit 4 when module vgr3.f and its associated subroutines are used. Note that the number of fields within each record may not match the FORMAT given because some of the FORMAT statements are used for other purposes with the subprograms.

Record number	Input variables	Format
1	RE	E15.8
2	NUMQ	I5
3	(NQUAD(I),I=1,NUMQ)	10I5
4	(NQUADA(I),I=1,NUMQ)	10I5
5*	(RQUAD(I),I=1,NUMQ-1)	9E15.8
6	LSTAP	L1
7	NLINE	I5
8	MES	A80
9	(IS(IJ,1),IJ=1,NDIM)	10I5
10	(IS(IJ,2),IJ=1,NDIM)	10I5
11	(JIN(IJ,1),IJ=1,NDIM)	10I5
12	(JIN(IJ,2),IJ=1,NDIM)	10I5
13	(JIN(IJ,3),IJ=1,NDIM)	10I5
14	IETA,IVIBJ,ISYMU,ICALC	10I5
15	NUMT1	I5
16	(IQ1(I),I=1,NUMT)	10I5
17	(IQ2(I),I=1,NUMT)	10I5
18	(IMU(I),I=1,NUMT)	10I5
19	RMS,OMEGA,RCENT,OMEGGH,LRCENT,SCAL2	4E15.7,L1,E15.7
20	NQWT,NBASIS,NPTSMX,IPRINT,INODE,IWGHT	6I5
21	NINT,NL,NSAVE,JSAVE,IHOM	5I5
22%	NPTS	I5
23@	NUMT2	I5

*Only if NUMQ=1
%Card repeated until NPTS=0, Only if NSAVE.NE.0
@Repeated NUMQ-1 times; not needed if NUMQ=1

11.4.4 EXPLANATION OF INPUT VARIABLES TO FORTRAN UNIT 4

Record 1: RE: (E15.8)

This variable is the equilibrium bond length of the diatom.

Record 2: NUMQ: (I5)

This variable is the number of different orders of quadrature (vibrational and/or angular) to be used during the propagation.

Record 3: (NQUAD(I),I=1,NUMQ): (10I5)

This variable is the number of points to be used in the vibrational quadrature for R less than RQUAD(I), where RQUAD (I) is discussed in Record 5 below. If one wishes to use a rigid rotor, NQUAD is set to 1, and the flag governing the use of a rigid rotor is set by NSAVE in Record 14 (see below).

Record 4: (NQUADA(I),I=1,NUMQ): (10I5)

This variable is the number of points to be used in the angular quadrature for R less than RQUAD(I), where RQUAD(I) is discussed in Record 5 below. Note that the separate specification of NQUAD and NQUADA allows the orders of the two quadratures to be varied independently.

Record 5: (RQUAD(I),I=1,NUMQ-1): (9E15.8) ONLY NEEDED IF NUMQ > 1

This variable governs the value of the propagation coordinate R such that the order of the vibrational and/or rotational quadrature will change for R .GT. RQUAD(I). See also Section 23.

Record 6: LSTAP: (L1)

This variable governs whether just one interaction potential is used. It is a logical variable ordinarily set equal to .TRUE.. It is used to tell RMPROP whether to do more than one approximation to the interaction potential during a run, i. e., if LSTAP .EQ. .TRUE., then the current potential is the only one used. If LSTAP .EQ. .FALSE., then following the end of the first run, RMPROP will cycle back and attempt to re-run subroutine PREPOT. Usually this means that the program will attempt to read parameters for a new potential from FORTRAN unit 4.

Record 7: NLINE: (I5)

This variable is a control variable used to define the number of lines of text to be read from FORTRAN unit 4 in Record 8.

Record 8: MES: (A80)

This variable consists of NLINE lines of text. These lines of text are ordinarily used to record details abount the individual test run or the

potential surfce being used.

Record 9: (ISJ(IJ,1),IJ=1,NDIM)): (10I5)
This variable contains the internal vibrational quantum numbers of the first diatom.

Record 10: (ISJ(IJ,2),IJ=1,NDIM)): (10I5)

This variable contains the internal vibrational quantum numbers of the second diatom.

Record 11: (JIN(IJ,1),IJ=1,NDIM)): (10I5)

This variable contains the internal rotation angular momentum quantum number of the first diatom.

Record 12: (JIN(IJ,2),IJ=1,NDIM)): (10I5)

This variable contains the internal rotation angular momentum quantum number of the second diatom.

Record 13: (JIN(IJ,3),IJ=1,NDIM)): (10I5)

This variable contains the internal angular momentum quantum number which is the vector sum of the internal rotational angular momenta of the first and the second diatoms.

Record 14: IETA,IVIBJ,ISYMU,ICALC: (4I5)

These variables have the following meanings:

IETA is the eigenvalue of the operator for exchange of identical diatoms with eigenvalue -1**(IETA).

IVIBJ is a switch which determines whether rotational coupling is included in the calculation of of the vibrational quadrature.
IVIBJ .EQ. 0 means use j-dependent vibrational states (rotational coupling).
IVIBJ .NE. 0 means use j-independent vibrational states (no rotational coupling).

ISYMU .EQ. 0 means use exchange symmetry; ISYMU .NE. 0 means do not use it.

ICALC governs the calculation of angular integrals.
ICALC .EQ. 0 means calculate angular integrals and continue.
ICALC .EQ. 1 means calculate angular integrals, and write them to UNIT 2
ICALC .EQ. 2 means read pre-existing angular integrals from UNIT 2 and continue program execution.
It is important to note that if the number of terms in the expansion of the interaction potential is decreased as a function of the propagation coordinate, then the existing angular integral matrix in FORTRAN unit 2 will be overwritten when the number of terms in the expansion is decreased. A subsequent run which tries to use the original size of the potential expansion will crash with a warning message

In ANGINT, the dimensions have changed ... , ...

when it tries to read from a copy of FORTRAN unit 2 which has been overwritten, where ... and ... are the number of words currently in FORTRAN unit 2 and the number of words requested by vgr3.

Record 15: NUMT1: (I5)

This variable gives the initial number of terms in the expansion of the interaction potential in Launay angular functions. If NUMQ on record 1 is greater than one, then after propagating past each RQUAD(I), new values for the number of terms in the angular expansion are read from record 23.

Record 16: (IQ1(I),I=1,NUMT): (10I5)

This variable gives the first index of the I'th Launay function in which the interaction potential is to be expanded.

Record 17: (IQ2(I),I=1,NUMT): (10I5)

This variable gives the first index of the I'th Launay function in which the interaction potential is to be expanded.

Record 18: (IMU(I),I=1,NUMT): (10I5)

This variable gives the first index of the I'th Launay function in which the interaction potential is to be expanded.

NOTE: The IQ1, IQ2 and IMU arrays will be reordered by the program in order to put the sum IQ1 + IQ2 in increasing order. This facilitates the dropping of terms later in the propagation, if NUMQ > 1.

Record 19: RMS,OMEGA,RCENT,OMEGGH,LRCENT,SCAL2: (4E15.7,L1,E15.7)

These variables are used by SUBROUTINE WEIGH which calculates the weights and nodes for the vibrational quadratures which compute the vibrational portion of the interaction potential matrix. This routine calculates the weights for the quadrature given the (user-supplied) diatomic potential curve (e. g. a harmonic oscillator or a Morse), by transforming the wave function into a harmonic oscillator basis and performing the integrals in that basis.

RMS is the reduced mass of the diatom in atomic units.

OMEGA is the angular frequency of the harmonic oscillator used in the basis.

RCENT is the equilibrium bond length of the oscillator.

OMEGGH is the frequency used to find scaled Gauss-Hermite nodes. If OMEGGH is zero, OMEGA is used instead.

LRCENT is a logical variable.
If LRCENT .EQ. .TRUE. calculate the value of the wave function at -RCENT.

SCAL2 governs scaling of the Gauss Hermite nodes.

Record 20: NQWT,NBASIS,NPTSMX,IPRINT,INODE,IWGHT: (6I5)

These variables are used by SUBROUTINE WEIGH which calculates the weights and nodes for the vibrational quadratures which compute the vibrational portion of the interaction potential matrix.

NQWT is the order of the quadrature used to evaluate matrix elements in the harmonic oscillator basis.

NBASIS is the number of harmonic oscillator states to include in the expansion.

NPTSMX is the maximum size of the vibrational quadrature to be found. This must be greater than or equal to the largest value of NQUAD on card 3.

IPRINT .GE. 0 for various debugging output to standard output. This should be used cautiously because it can generate large quantities of output.

INODE determines the type of nodes used:

INODE= 1 use semi-classical nodes.
INODE= 2 use scaled Gauss-Hermite nodes.
INODE= 3 use Gaussian nodes for the weight function PSI(IWGHT)**2.
INODE= 4 call SDATA to get other (user-supplied) nodes.

It should be noted in passing that although SUBROUTINE WEIGH has been tested with all values of INODE, the elements of the test suite included in RMPROP-Version 2.1.2 only include the values 1 and 3. In addition, the user should supply a dummy version of SDATA for INODE = 1, 2, or 3 to prevent error messages on loading.

IWGHT runs from 1 upwards, and is used in conjunction with INODE .EQ.3.

Record 21: NINT,NL,NSAVE,JSAVE,IHOM: (5I5)

These variables are used by SUBROUTINE WEIGH which calculates the weights and nodes for the vibrational quadratures which compute the vibrational portion of the interaction potential matrix.

NINT and NL only have meaning when INODE .EQ. 1 (semiclassical nodes).

NINT is the number of intervals to be used by SUBROUTINE NODE, which calculates semi-classical nodes.

NL is the order of Legendre interpolation used by SUBROUTINE NODE, which calculates semiclassical nodes.

NSAVE is the maximum vibrational quantum number + 1 for which weights and nodes are to saved on FORTRAN unit 1. If NSAVE is 1, special output is generated for a rigid rotor.

JSAVE is the maximum rotational quantum number - 1 of rotational states in the calculation of the vibrational states. If no rotational-vibrational coupling is desired, use JSAVE=0.

IHOM should be zero for a heteronuclear diatom, 1 for a homonuclear diatom with only odd rotational states or 2 for a homonuclear diatom with only even rotational states.

Record 22: NPTS: (I5) Only if NSAVE.NE.0; CARD REPEATED AT LEAST FOR EACH VALUE OF NQUAD UNTIL 0 READ IN

This variable is used by SUBROUTINE WEIGH which calculates the weights and nodes for the vibrational quadratures which compute the vibrational portion of the interaction potential matrix. No value of NPTS can be larger than NPTSMX, and each value of NQUAD (see record 3) must be a value of NPTS. The final NPTS value must be zero.

Record 23: NUMT2: (I5) CARD REPEATED NUMQ-1 TIMES

This variable is read in NUMQ-1 times by POT in vgr3.f, whenever a value of RQUAD(I) (see Record 5 above) is passed during the propagation. NUMT2 is the new number of terms in the Launay expansion for values of the propagation coordinate between RQUAD(I) and RQUAD(I+1) (for I less than NUMQ - 1), and for values of the propagation coordinate greater than RQUAD(I) (for I equal to NUMQ). Note that the angular integral matrix discussed in Section 11.4.1 above will be re-written to allow for the value of NUMT.

12. DESCRIPTION OF INPUT/OUTPUT FILES

This section describes the various input/output files used by the main program. The files are accessed in the FORTRAN program by means of OPEN and CLOSE statements, which refer to the files as 'fort.X' where X is the unit number given in the first column below. However, depending on the options used, not all of the files may be required for a given run. In addition to these units, in our test suite, subprograms PREPOT and POT utilize FORTRAN units 1, 2, and 4. If the user wishes to assign logical names to these files, this must be done in the command stream used to submit the job. For example, the command stream for test run 19.13, "hei2r1.csh", assigns FORTRAN unit 4 and FORTRAN unit 5 as follows:

```
/bin/cp ${TDIR}/hei2r1.4 fort.4
/bin/cp ${TDIR}/hei2r1.5 fort.5
```

Unit	Input or Output	Usage
3	I or O	Contains date and time of current run
5	I	Input to main program.
6	O	Standard output.
7	O	Symmetrized reactance matrix.
8	I or O	TAUE matrix and perhaps its inverse {8.7, 8.9}. (nonASCII)
10	O	Transition matrix elements.
11	O	Eigenphase sum of reactance matrix and r for each asymptotic analysis (individual eigenphases and eigenvectors are not written to this file, but are written to FORTRAN unit 16).
12	I or O	Sector midpoints and stepsizes.
14	I or O	Data saved by first case of type-1 second-case runs (needed to begin second case of type-1 second-case runs at a later date {8.9}). (nonASCII)
15	O	Short output of diagonal elements and first row of scattering matrix for each energy/basis either at convergence or RMAX, as appropriate.
16	O	RABC(IS) values and eigenphases and eigenvectors of reactance matrix for each asymptotic analysis.
17	I or O	Restart information to continue current propagation at later date. (nonASCII)
18	I or O	Alternate restart information to continue current propagation at later date. (nonASCII)
> or =19	O	Diagnostic information at a specific sector selected by the value of the input variable ISAVV.

13. INPUT TO MAIN CODE

13.1 SUMMARY

A record-by-record summary of the input to the RMPROP code from FORTRAN unit 5. Note that the number of fields within each record may not match the FORMAT given because some of the FORMAT statements are used for other purposes with the subprograms.

Record number	Input variables	Format
1	TITLE1	A72
2	TITLE2	A72
3	LJRI, LR4MT	2L1
4	ID, NSECT, NAN, NSAVE, NVIB, NVIB1, NNE	7I5
5	IREST, ISAVE, ISAVV, IUNXX, IFTXX	10I5
6	MUABC, NDIM, JTOT	G15.6,2I5
7	L(I), I = 1, NDIM	10I5
8	NPD, NE, IROWS	3I5
9	PD(I), I = 1, NPD	10I5
10	E(I), RSTART(I), OAM(I), I = 1, ABS(NE)	2G15.6,I5
11	H(1), H(2), RMAX	3G15.6
12	NEPS	I5
13	EPSA(I), I = 1, NEPS	5G15.6
14	EPSB(I), I = 1, NEPS	5G15.6
15	EPSC(I), I = 1, NEPS	5G15.6
16	HMINA(I), I = 1, NEPS	5G15.6
17	HMINB(I), I = 1, NEPS	5G15.6
18	HMINC(I), I = 1, NEPS	5G15.6
19	HMAXA(I), I = 1, NEPS	5G15.6
20	HMAXB(I), I = 1, NEPS	5G15.6
21	HMAXC(I), I = 1, NEPS	5G15.6
22	REPSA(I), I = 1, NEPS	5G15.6

23	REPSB(I), I = 1, NEPS	5G15.6
24	NCUSP	I5
25	RCUSP(I), I = 1, NCUSP	5G15.6
26	EPSRED, RBIG, NPROP	2G15.6,I5
27	LPSINF, LPSYM(I), I = 1, 8, LOPUT	10L1
28	LPR(I), I = 1, 10	10L1
29	EPSDR, RASYE	2G15.6
30	LBES, LTASY, LPEPS, IMATCH, ITPRNT, ITCOL	3L1,3I5
31	IPMX	I5
32	RPSM(I), I = 1, IPMX	5G15.6
33	LGS(I), I = 1, 15	20L1
34	LBGS(I), I = 1, 15	20L1
35	IOPT2, IOPT3	2I4
36	EPSMAG, EPSPH	2G16.5
37	NIND	3I4
38	IND(I,1), I = 1, MIN(NIND,10)	10I4
39	IND(I,2), I = 1, MIN(NIND,10)	10I4

13.2 EXPLANATION OF INPUT VARIABLES

Record 1: TITLE1: (A72)

This variable is a line read in to describe the attributes of the specific run (molecular system, potential, total angular momentum, energy). Only the first 72 characters are used.

Record 2: TITLE2: (A72)

This variable is a line read in to describe the purpose of the specific run (e. g. test suite, convergence with respect to stepsize, etc.). Only the first 72 characters are used.

Record 3: LJRI, LR4MT: (2L1)

LJRI: if this variable is .TRUE. the run will just read and echo the contents of FORTRAN unit 5 and stop with message 5099. If it is .FALSE. the run continues normally.

LR4MT: this variable controls the (in)homogeneous option {8.5, 8.6}. If it is .TRUE., then the homogeneous option {8.5} has been selected (preferable for heavy-body collisions). If it is .FALSE., then the inhomogeneous option has been selected.

Record 4: ID, NSECT, NAN, NSAVE, NVIB, NVIB1, NNE: (7I5)

These variables are set to the input values for the current run. These variables correspond to the contents of the PARAMETER statement which is contained in the file "parameter.inc" which is discussed in Sections 14 and 15 below. The values in parameter.inc are values used to dimension arrays in many subroutines of the program, and may be larger than the values given here.

Record 5: IREST, ISAVE, ISAVV, IUNXX, IFTXX: (10I5)

These variables control the usage of restart information. For large calculations, it may be advisable to save intermediate information which can be used to restart the calculation with minimal loss of resources in the event of computer failure. These options are also useful to break up a large calculation into smaller pieces and to check intermediate results.

IREST: if nonzero, this becomes a restart run. The program will attempt to read in restart information (written by an earlier run with the same size basis set and energy) from either FORTRAN unit 17 or FORTRAN unit 18 depending upon the value of IREST. If IREST = 17 then the program reads from FORTRAN unit FORTRAN unit 17; if IREST = 18 then the program reads from FORTRAN unit 18, and if IREST has any other nonzero vale then the program issues a warning message and attempts to read from FORTRAN unit 17. In order to restart successfully, ISAVE must have been nonzero in the original run.

ISAVE: restart information will be saved every ISAVE sectors alternating between FORTRAN unit 17 and FORTRAN unit 18 if ISAVE is greater than zero.

ISAVV: This is the maximum number of sectors to propagate in this run. That is, when sector number ISAVV is reached, diagnostic information is saved and the program stops. The program cannot be restarted from this information. The output generated by this option is only meant to be diagnostic in order to make sure that the program is running correctly before proceeding with a large run. If ISAVV is read in as 0, this option is disabled.

IUNXX: used in conjunction with ISAVV. IUNXX denotes the logical unit to which diagnostic information should be written when IS (the sector number) equals ISAVV (not ISAVE!). If an invalid unit number is chosen (less than 19) then IUNXX is set to 19; if IUNXX is set to 0 then information is written to standard output (FORTRAN unit 6).

IFTXX: used in conjunction with IUNXX and ISAVV. If nonzero the output is written in binary (unformatted) form, if zero the output is written in formatted form. (If output is written to FORTRAN unit 6, this is overridden, and output will be written in formatted form regardless of IFTXX).

Record 6: MUABC, NDIM, JTOT: (G15.6,2I5)

MUABC: the reduced mass for the system {8.2}.

NDIM: the number of close coupled equations to be solved {8.2}. NDIM must be less than or equal to the PARAMETER ID.

JTOT: the total angular momentum for the run. {8.2}

Record 7: L(I), I = 1, NDIM: (10I5)

L(I): the orbital angular momentum for the I'th channel {8.2}.

Record 8: NPD, NE, IROWS: (3I5)

NPD: the number of contracted basis set dimensions to run. NPD must be less than or equal to 10. Note that only one of NPD, NE, or NEPS can be greater than 1 in a given run.

NE: indicates the number of multiple energy/multiple orbital angular momentum/multiple starting point runs to be performed {8.9}. If positive then NE energies will be run using type-1 second energies. If negative, -NE energies will be run using type-2 second-case runs. Note that only one of NE, NPD, and NEPS can be different from 1 at any time.

IROWS: a flag to control reading or writing propagation information {8.9} and type-1 second-case data.

IROWS = -1 to perform a type-1 second-case run used data files generated in a separate run. This requires the files FORTRAN unit 8, and FORTRAN UNIT 14, which is produced using IROWS = 1 or IROWS = 4. In this case NE should be greater than 1 and the first energy, first RSTART and first OAM read in will be ignored (see record 7).

IROWS = 0 to perform a single energy run or a multiple energy run in which no data is saved for later-date second-case runs.

IROWS = 1 to save the necessary data on FORTRAN unit 8 and FORTRAN unit 14 for a later-date second-case run.

IROWS = 2 to write the sector midpoints and stepsizes to FORTRAN unit 12.

IROWS = 3 to read the sector midpoints and stepsizes from FORTRAN unit 12 and use them from the 3rd sector onwards.

IROWS = 4 reads stepsizes like IROWS = 3 and writes second-case information like IROWS = 1.

Record 9: PD(I), I = 1,NPD: (10I5)

PD(I) is the starting size of the I'th contracted basis set {8.7}. The program does a scattering calculation for I = 1, NPD. Note that PD(I) must be at least 1 and at most NDIM.

Record 10: E(I), RSTART(I), OAM(I), I = 1, ABS(NE): (2G15.6,I5)

E(I): the total energy of the I'th multiple-case run {8.8}. Recall that if IROWS = -1, then E(1) is not used.

RSTART(I): the center of the first sector for the I'th multiple-case run {8.4}. The smallest value of RSTART(I) must be RSTART(1). Also note that RSTART(I)-H(1) must be greater than zero. If IROWS = -1, RSTART(1) is not used.

OAM(I): the orbital angular momentum in every channel for the I'th multiple-angular-momentum run {8.8}. If any of the OAM(I) are nonzero, then all of the L(I) (see input record 7 above) must be zero. If IROWS = -1, OAM(1) is not used.

Record 11: H(1), H(2), RMAX: (3G15.6)

H(1): the stepsize in the first sector {8.4}.

H(2): the stepsize in the second sector {8.4}.

RMAX: the maximum integration distance {8.4, 8.15}.

Record 12: NEPS: (I5)

NEPS: the number of stepsize criteria to be used. The program does a separate scattering calculation using each of the NEPS values of the stepsize parameters. NEPS must not be greater than 5 and if it is greater than 1, only single-energy and single-basis-set runs are possible. Note that only one of NPD, NE, or NEPS can be greater than 1 in a given run.

Record 13: EPSA(I), I = 1, NEPS: (5G15.6)

EPSA(I): the variable stepsize criterion for r less than REPSA(I) {8.4}. A typical value is 0.1.

Record 14: EPSB(I), I = 1, NEPS: (5G15.6)

EPSB(I): the variable stepsize criterion for r greater than or equal to REPSA(I) but less that REPSB(I) {8.4}. A typical value is 0.1.

Record 15: EPSC(I), I = 1, NEPS: (5G15.6)

EPSC(I): the variable stepsize criterion for r greater than or equal to REPSB(I) {8.4}. A typical value is 0.1.

Record 16: HMINA(I), I = 1, NEPS: (5G15.6)

HMINA(I): the minimum stepsize for r less than REPSA(I) {8.4}. A typical value is 0.001 to 0.1 bohr.

Record 17: HMINB(I), I = 1, NEPS: (5G15.6)

HMINB(I): the minimum stepsize for r less than REPSB(I) {8.4}.

Record 18: HMINC(I), I = 1, NEPS: (5G15.6)

HMINC(I): the minimum stepsize for r less than REPSA(I) {8.4}.

Record 19: HMAXA(I), I = 1, NEPS: (5G15.6)

HMAXA(I): the maximum stepsize for r less then REPSA(I) {8.4}. Typically HMAXA is large enough so that the stepsize is unlimited; or is set equal to HMINA(I) to ensure fixed stepsizes.

Record 20: HMAXB(I), I = 1, NEPS: (5G15.6)

HMAXB(I): the maximum stepsize for r less then REPSB(I) {8.4}. Typically HMAXB is large enough so that the stepsize is unlimited; or is set equal to HMINB(I) to ensure fixed stepsizes.

Record 21: HMAXC(I), I = 1, NEPS: (5G15.6)

HMAXC(I): the maximum stepsize for r less then REPSA(I) {8.4}. Typically HMAXC is large enough so that the stepsize is unlimited; or is set equal to HMINC(I) to ensure fixed stepsizes.

Record 22: REPSA(I), I = 1, NEPS: (5G15.6)

REPSA(I): the distance which demarcates the end of the first stepsize criterion region {8.4}.

Record 23: REPSB(I), I = 1, NEPS: (5G15.6)

REPSB(I): the distance which demarcates the end of the second stepsize criterion region {8.4}.

Record 24: (NCUSP): (I5)

NCUSP: the number of cusps in the potential {8.4}. A cusp is a place where the potential is continuous but has discontinuous first derivatives.

Record 25: RCUSP(I), I = 1, NCUSP : (5F15.6) (Include even if NCUSP = 0)

RCUSP: the location of the I’th cusp {8.4}.

Record 26: EPSRED, RBIG, NPROP: (2G15.6,2L1)

EPSRED: the criterion to drop uncoupled channels from the propagation {8.7}. The program will drop channels only when r is greater than RBIG. Only asymptotically closed channels are dropped. A typical value is 10^-6.

RBIG: the distance where the program will first test to see if it can drop channels from propagation {8.7}. The program will only drop channels for the homogeneous option {8.5}.

NPROP: governs mode of propagation {8.10}.

If NPROP = 0 then explicitly calculate sector r matrices and use inverse of overlap matrix.

If NPROP = 1 then explicitly calculate sector r matrices and use transpose of overlap matrix.

If NPROP = 2 then do not explicitly calculate sector r matrices and use inverse of overlap matrix.

NPROP = 2 is the fastest of the options, and NPROP = 1 (the transpose of the overlap matrix) is not rigorous for PDIM less than NDIM, i. e. where channels are dropped from propagation. NPROP = 2 is valid only for the homogeneous option {8.5}.

Record 27: LPSINF, LPSYM(I), I = 1, 8, LOPUT: (10L1)

LPSINF: if .TRUE., for r greater than RBIG, in every sector the sector number, sector midpoint and EFACT (the sum of the elements of the first row of the potential matrix, less the diagonal, if available) will be printed.

LPSYM(1): if .TRUE., test the symmetry of the R^4 matrix at each sector for each energy just before entering subroutine RPROP.

LPSYM(2): if .TRUE., test the orthogonality of the TAUE matrix {8.7} for each energy just after initial call to subroutine TAUMTS, for all r > RBIG. This option only works for LTYPE2 second case runs.

LPSYM(3): if .TRUE., test the symmetry of the R^4 matrix at each sector after call to subroutine TAUMTS and before stepsize determination.

LPSYM(4): if .TRUE., upon entrance to subroutine GNSCAT, calculate the maximum relative difference and the average relative difference of the off-diagonal elements, and print out the R matrix in the mixed basis {8.11}.

LPSYM(5): if .TRUE., print out the R matrix transformed to the asymptotic basis {8.11}, and print out the maximum relative difference and average relative difference of the off-diagonal elements.

LPSYM(6): if .TRUE., print out the maximum relative difference and the average relative difference of the off-diagonal elements of the reactance matrix, and print out the reactance matrix itself.

LPSYM(7): if .TRUE., in subroutine RPROP, check the symmetry of the R matrix every five sectors and print out the sum of the first row and trace.

LPSYM(8): if .TRUE, in subroutine GNSCAT, print out the matrix EFASY from subroutine POT, the matrix of eigenvectors at the most recent sector, and the matrix which transforms from the current sector basis to the asymptotic basis. It also prints out the sums of the squares of all elements of these matrices, and information such as the current number of channels propagated, and the number of asymptotically open channels at the current energy. This generates a great deal of output and should only by used if necessary to help debug a new potential energy surface.

LOPUT: if .TRUE., some information concerning the output is printed.

Record 28: LPR(I), I = 1, 10: (10L1)

LPR(1): if .TRUE., print execution times of various tasks.

LPR(2): if .TRUE., print asymptotically diagonalized D {8.7} matrix at all energies.

LPR(3): is not used.

LPR(4): if .TRUE., print the interaction matrix at each sector at the first energy.

LPR(5): if .TRUE., print the eigenvalues at the center of each sector at all energies.

LPR(6): if .TRUE., print the eigenvectors at the center of each sector at all energies.

LPR(7): if .TRUE., print the TAUE {8.7} matrix and its inverse (TINVE) at each sector.

LPR(8): if .TRUE., print the sector r matrices at each sector.

LPR(9): if .TRUE., print the global R_4 matrix at each sector for the homogeneous option {8.5}, and print the global R matrix at each sector for the inhomogeneous option {8.6}.

LPR(10): if .TRUE., print the sector midpoints and stepsizes.

Record 29: EPSDR, RASYE: (2G15.6)

EPSDR: the criterion to drop closed channels in the asymptotic analysis {8.12}. A typical value is 0.001.

RASYE: a distance where the potential has reached its asymptotic value, the distance at which various quantities needed for the large-R boundary conditions will be evaluated {8.13}.

Record 30: LBES, LTASY, LPEPS, IMATCH, ITPRNT, ITCOL: (3L1,3I5)

LBES: if .TRUE., the program will match to Ricatti-Bessel functions {8.13}. Otherwise, sine/cosine functions will be used.

LTASY: if TRUE., prior to performing the asymptotic analysis, the channels are transformed into the asymptotic basis using the array EFASY supplied by (the user-supplied subroutine) POT {8.11}. This procedure requires that all of the open channels occur before all of the closed channels.

LPEPS: if .TRUE., the eigenphase sum will be calculated and the results written to FORTRAN unit 6.

IMATCH: specifies the wavevectors used in the asymptotic analysis {8.13}. One normally uses IMATCH = 0.

IMATCH = 0: use the asymptotic wave vectors, evaluated at RASYE.

IMATCH = 1: use the eigenenergies calculated at the middle of the current sector.

IMATCH = 2: use the eigenenergies calculated at the outer edge of the current sector.

ITPRNT: the number of rows of the T matrix to write to FORTRAN unit 10.

ITCOL: the number of columns of the T matrix to write to FORTRAN unit 10. ITPRNT and ITCOL are separate options from LGS(14) which also saves the T matrix on FORTRAN unit 10.

Record 31: IPMX: (I5)

IPMX: the number of intermediate distances at which the asymptotic analysis is carried out. IPMX must be less than or equal to 20. Depending on the values of LGS(2) and IOPT2, the asymptotic analysis at some intermediate distances may not print out any information, but the real and imaginary portions of the scattering matrix at the first energy {8.15} will be saved for comparison.

Record 32: RPSM(I), I = 1, IPMX: (5G15.6) (Include even if IPMX = 0)

RPSM(I): an intermediate distance at which the asymptotic analysis will be performed {8.15}. RPSM(I) should be greater than RBIG. The program performs the asymptotic analysis at the first sector for which RABC(IS).GE.RPSM(I), except if the previous asymptotic analysis (RPSM(I-1)) was performed at the previous sector (I-1). In the latter case, the present intermediate distance asymptotic analysis is skipped.

Record 33: LGS(I), I = 1, 15: (20L1)

LGS(1): if .TRUE., then each time the scattering matrix is printed the program will also print RABC(i), the midpoint of the sector, and EFACT, which is the sum of the off-diagonal elements of the first row of the potential matrix.

LGS(2): if .TRUE., print the scattering matrix one or more times based on the following options by setting the integer input variable IOPT2 = j.

j	option chosen
(1)	each time it is calculated
(2)	for the previous and current sectors after the calculations converge: this choice will lead to errors unless the asymptotic analysis is performed at least twice, for example once when RABC(IS) is less than RMAX and at the first sector for which RABC(IS) is greater than or equal to RMAX.
(3)	in the last sector only (determined by the value of NSECT or of RMAX).

LGS(3): if .TRUE., check unitarity and print checks each time the scattering matrix is printed.

LGS(4): if .TRUE., in the last sector calculate and print the phase shifts.

LGS(5): if .TRUE., in the last sector calculate and print the cross sections. (These need to be summed over JTOT {8.2} to yield the observable integral cross sections).

LGS(6): if .TRUE., in the last sector calculate and print the transition probabilities.

LGS(7): if .TRUE., in the last sector print the reactance matrix.

LGS(8): if .TRUE., in the last sector try to calculate and print the quantities requested in LGS(4), LGS(5), and LGS(6) even if the calculations calculations are terminated before convergence is reached: this option will also work if the asymptotic analysis is not performed at any intermediate distances.

LGS(9): if .TRUE., calculate the symmetrized reactance matrix select one of the following options by setting the integer input variable IOPT3 equal to j.

option (j)
- (1) symmetrize the reactance matrix by imposing the lower left triangle on the upper right.
- (2) symmetrize the reactance matrix by imposing the upper right triangle on the lower left.
- (3) symmetrize the reactance matrix by setting rscat(i,j) = rscat(j,i) equal to the arithmetic mean of the originally calculated off diagonal elements.
- (4) each of the above in turn

LGS(9) must be true if any of the following are true:

LGS(10): if .TRUE., in the last sector print the symmetrized reactance matrix.

LGS(11): if .TRUE., in the last sector print the symmetrized scattering matrix and check unitarity.

LGS(12): if .TRUE., from the symmetrized matrices calculate and print the quantities requested in LGS(4), LGS(5),and LGS(6).

LGS(13): if .TRUE., write the symmetrized reactance matrix onto FORTRAN unit 7.

LGS(14): if .TRUE., write JTOT (the total angular momentum), the energy and the transition matrix to FORTRAN unit 10 using (1X,I5,1PE15.8) and (1X,1PE13.6) format. Subroutine EXSCAT will do this.

LGS(15): if .TRUE., calculate and write the eigenphases. The eigenphases as a function of r will be written with r on FORTRAN unit 11, and the eigenphases, r and the eigenvectors will be written on FORTRAN unit 16.

Record 34: LBGS(I), I = 1,15: (20L1)

LBGS specifies what is printed when a value of RPSM is reached. The LBGS(I) have the same meaning as the LGS(I).

Record 35: IOPT2, IOPT3: (3I5)

IOPT2: used to determine if and when the scattering matrix will be printed out during the asymptotic analysis. The options are as follows:

IOPT2: Scattering matrix will be printed:

(1) each time it is calculated
(2) for the previous and current sectors after the calculations converge: this choice will lead to errors unless the asymptotic analysis is performed at (at least) one intermediate distance.
(3) in the last sector only

IOPT3 determines the method used to symmetrize the reactance matrix {8.14}. The allowed values for IOPT3 are:

IOPT3: Method of symmetrization employed

(1) symmetrize the reactance matrix by imposing the lower left triangle on the upper right.
(2) symmetrize the reactance matrix by imposing the upper right triangle on the lower left.
(3) symmetrize the reactance matrix by setting rscat(i,j) = rscat(j,i) equal to the arithmetic mean of the originally calculated off diagonal elements.
(4) each of the above in turn

Record 36: EPSMAG,EPSPH: (2G16.5)

EPSMAG: The convergence criterion for the magnitude of the relative convergence of the scattering matrix elements {8.15}, if it is nonzero. If it is zero then the magnitudes are not tested.

EPSPH: The convergence criterion for the magnitude of the relative convergence of the phase of the scattering matrix elements {8.15}, if it is nonzero. If it is zero then the phases are not tested.

The remaining records are read in if and only if either EPSMAG or EPSPH or both are nonzero.

Record 37: NIND : (3I4)

NIND: the number of individual elements of the scattering matrix which will be tested for convergence with respect to increasing rmax {8.15}.

Record 38: IND(I,1), I = 1, MIN0(NIND,10): (10I4)

IND(I,1): an array containing the channel numbers for the initial states of the scattering matrix elements to be tested for convergence with respect to increasing rmax {8.15}.

Record 39: IND(I,2), I = 1, MIN0(NIND,10): (10I4)

IND(I,2): an array containing the channel numbers for the final states of the scattering matrix elements to be tested for convergence with respect to increasing rmax {8.15}. It should be emphasized that IND(I,1) and IND(I,2) refer to channels and not to internal quantum numbers. For example, if NIND were 1, and IND(1,1) = 3 and IND(1,2) = 4, then the only element of the scattering matrix to be tested for convergence would be (3,4).

14. ARRAY DIMENSIONS AND MEMORY CONSIDERATIONS

There are seven parameters specifying the amount of memory allocated in this program. They are:

IDX: must be greater than or equal to the maximum number of primitive basis functions {8.2}.

NSECTX: must be greater than or equal to the maximum number of sectors {8.4}.

NANX: must be greater than or equal to the maximum number of angular momentum quantum numbers per channel, excluding the orbital angular momentum for relative motion.

NSAVEX: must be greater than or equal to the number of TAUE {8.7} matrices and their inverses saved in memory for type-1 second-case runs {8.9}.

NVIBX: must be greater than or equal to the maximum number of vibrational quantum numbers per channel.

NVIB1X: must be greater than or equal to the average amount of space per channel pair to allocate for the array BPOT for use by SUBROUTINE POT.

NNEX: must be greater than or equal to the maximum number of second energies for type-2 second-case runs {8.9}.

NBARX: has the value IDX*(IDX+1)/2. Used to dimension the array PCDMT used by the alternative eigensystem analysis routine EV2RSP.

All parameters are set in "parameter.inc" which is passed to all subroutines where needed by the FORTRAN statement INCLUDE "parameter.inc".

The memory of the program is primarily composed of "big" arrays: those on the order of (IDX,IDX). A summary of the largest arrays and a brief description of the contents of each follows:

EFCNM1(IDX,IDX): stores transpose of the eigenvectors of the D matrix {8.11}.

BPOT(IDX,IDX,NVIB1X): stores data for SUBROUTINE POT.

R1MT(IDX,IDX,NNEX): global R_1 matrix for each type-2 second-case run.

R2MT(IDX,IDX,NNEX): global R_2 matrix for each type-2 second-case run.

R3MT(IDX,IDX,NNEX): global R_3 matrix for each type-2 second-case run.

R4MT(IDX,IDX,NNEX): global R_4 matrix for each type-2 second-case run.

EFASY(IDX,IDX): stores eigenvectors in asymptotic basis.

TEMPBC(IDX,IDX*2): temporary storage array.

TEMP2(IDX,IDX): temporary storage array.

TEMP3(IDX,IDX): temporary storage array.

TAUE(IDX,IDX,NSAVEX): stores overlap matrix {8.7}.

TINVE(IDX,IDX,NSAVEX): stores inverse of overlap matrix.

SPRRE(IDX,IDX): stores real elements of scattering matrix at previous intermediate distance.

SPRIM(IDX,IDX): stores imaginary elements of scattering matrix at previous intermediate distance.

PCDMT(NBARX): packed interaction matrix for subroutine EV2RSP.

The arrays R4MT, EFCNM1, and BPOT must exist at all times.
The arrays R1MT, R2MT, and R3MT must exist if the inhomogeneous option {8.6} is selected (LR4MT = .FALSE.).
The array EFASY must exist at all times if LTASY is .TRUE. (it usually is).
The arrays SPRRE and SPRIM must exist at all times if convergence with respect to distance is being performed according to EPSMAG and EPSPH.
If NPROP = 1 or 2, TINVE may be the same as TAUE.
If no more than one reactance matrix symmetrization option is used in subroutine GNSCAT, then TEMP3 can be the same as TEMP2.
If NPROP = 2, and TYPE2 second energies are used, TEMP2 can be the same as TAUE. NPROP will be set to zero for LR4MT = .FALSE..
If LR4MT = .TRUE. and NPROP = 2, then R1MT, R1MTM1, R2MT, R2MTM1, R3MT, and R3MTM1 can be set equal to R4MT.
For large runs, use NSAVE=1.
Set NNEX = 1 unless the current run is a second-case run.
For large IDX, the current memory requirements are essentially

IDX*(26+NANX+NSECTX+2*NNEX+2*NVIBX+IDX*(10+NVIB1X+2*NNEX+2*NSAVEX))

if the homogeneous option {8.5} is selected, and

IDX*(26+NANX+NSECTX+2*NNEX+2*NVIBX+IDX*(10+NVIB1X+8*NNEX+2*NSAVEX))

if the inhomogeneous option {8.6} is selected.
The program is efficient in its use of memory, striving to attain a balance between low memory requirements and ease of usage. The memory requirements for the executable images of the test suite vary between 168707 and 203796 64 bit words on the Cray-2, including the code (120080 - 1718032 words), the initialized data (7488 - 9603 words), and the uninitialized data (58951 - 261075 words).

The figures below include the modules matrix.f, evdrsp.f, and sdrcray.f.

	code	initialized data	uninitialized data	total
hfhfstr1:	1718032	7518	311246	2036796
hehfr1:	120592	7488	58951	187031
heh2r1:	140048	7488	111285	258821

hfhfpbsr1:	143120	9603	163554	316277
hfhfadr1:	1580752	7526	101183	1689461
atdir1:	120080	7490	103913	231483
ehydr1:	120240	7488	40979	168707
hei2r1:	1579760	7564	261075	1848399

where the code is comprised of the machine-language version of the source deck, the initialized data is the data assigned values before run-time (e. g. in BLOCK DATA or PARAMETER statements), and the uninitialized data is data which is only assigned a value during execution.

15. DESCRIPTION OF LABELED COMMON BLOCKS AND PARAMETER LISTS

The program uses four labeled common blocks and one parameter statement to facilitate the passing of information. They are

```
      COMMON/PTIMCM/PARTIM(10)

      COMMON/LCM/LEFACT,LBES,LCONV,LEARLY,LENTER,LR4MT,LSTOP1,LSTOP2

      COMMON/TIMCOM/TIMC(50)

      COMMON/PRNTCM/IUNITS,IFTXX

      PARAMETER(IDX=ii1, NSECTX=ii2, NANX=ii3, NSAVEX=ii4,NVIBX=ii5,NVIB1X=ii6,
     1 NNEX=ii7,NBARX=(IDX*IDX+1)/2))
```

where the letters 'ii1, ii2, ...' in the PARAMETER statement are used to indicate integers which may vary from one test run to another according to the system studied.

Of potential interest to the user are the provision of timings to be passed through TIMCOM, and the elements of the PARAMETER statement. The entries in the parameter statement are used to provide run-specific dimensions to arrays so that the arrays are only as large as required for a given element of the test suite.

In the PARAMETER list above, the values passed have the following functions:

IDX is used to dimension many of the arrays in the program including R4MT, JIN, TEMPBC and others. IDX is the size of the close coupling expansion of the individual run.

NSECTX is used to dimension arrays such as RABC whose size depends upon the maximum number of sectors allowed by the program.

NANX is generally used to dimension arrays containing internal rotational quantum numbers for the collision partners.

NSAVEX is described in Section 14 of this manual.

NVIBX denotes the maximum number of the collision partners which are allowed to vibrate. If NVIB = 1 in FORTRAN unit 5, or if NVIBX is 1 in parameter.inc, then subroutine SUMS is inoperative.

NVIB1X is used to increase the size of array BPOT if necessary. BPOT is dimensioned (IDX,IDX,NVIB1X).

NNEX is the maximum number of second-case runs. For the test suite as supplied, this is two. If the user wishes to do a second-case run with more than two energies/angular momenta/starting points then this value should be increased in the PARAMETER statement; this may be done by modifying the file parameter.inc (or the file which is copied to parameter.inc by the C shell script executing the user's program).

NBARX - the dimension of the array PCDMT used by the alternative

eigensystem analysis routine EV2RSP; it has the value IDX*(IDX+1)/2

In COMMON/TIMCOM/, the values passed have the following functions:

TIMC(50) is an array used to hold timings of various tasks performed by the program. If LSTAP is .FALSE. then all the elements of the array TIMC are reset to zero before each new potential.

TIMC(1) is a timing of reading input from FORTRAN unit 5.

TIMC(2) is a timing of calls to subroutine POT.

TIMC(3) is a timing of the rephasing (if desired) of the eigenfunctions of the transformation matrix. This rephasing should not occur if LTASY is .TRUE..

TIMC(4) is a timing of calculating stepsizes for each sector.

TIMC(5) is a timing of saving the previous sector’s D matrix.

TIMC(6) is a timing of dropping channels from the asymptotic analysis.

TIMC(7) is a timing of calculating the S matrix.

TIMC(8) is a timing of subroutine EXSCAT.

TIMC(9)-TIMC(16) are timings of the checks of LPSYM(1)-LPSYM(8) respectively. (The WRITE statement for these is commented out.)

TIMC(17) is a timing of the matrix utility routine DGETRF (MINV).

TIMC(18) is a timing of the matrix utility routine DGEMM.

16. SUBPROGRAM DESCRIPTIONS

This portion of the manual contains a description of the main program and of subroutines called by the main program and its subroutines (except subroutines PREPOT, POT, and HEADER, which are described in Section 10, and the matrix utility routines, which are described in Section 17). This section is divided into three subsections. The first subsection contains a description of the main program and of those FORTRAN subroutines called by the main program to perform arithmetical tasks, regardless of the version of the program. The remaining subsections contain descriptions of the system dependent utility routines, "sdrcray.f", "sdrrs6000.f", "sdrsun.f", and "sdralpha.f".

16.1 MAIN PROGRAM AND FORTRAN SUBPROGRAMS

PROGRAM RMPROP: driver or main program, reads in control data for the run and physical information concerning the system and calls other subroutines. Execution stops here on normal completion.

SUBROUTINE DROP: drops channels which are uncoupled from all other channels from the propagation and closed channels from the asymptotic analysis {8.7, 8.12}.

SUBROUTINE EXSCAT: calculates phase shifts, contributions to the state-to-state cross sections from the value of the total angular momentum being considered, transition matrix elements and transition probabilities.

SUBROUTINE GNSCAT: calculates scattering and reactance matrices, and from these calculates other variables according to contents of logical array LGS.

SUBROUTINE MATCH: calculates the square of the local wave vector and the number of open channels at the local r {8.13}.

SUBROUTINE PRNTM2: called by many other routines to print out elements of square matrices in a particular fashion.

SUBROUTINE PRNTSU: prints out complex valued scattering matrix and tests its unitarity.

SUBROUTINE PRSCAT: reads in logical and control variables used in the asymptotic analysis (subroutine GNSCAT).

SUBROUTINE RBES: calculates regular and irregular Ricatti-Bessel functions and their derivatives.

SUBROUTINE RPROP: calculation of sector r matrices, and global R_4 matrix for the homogeneous option {8.5}, and global R matrix for inhomogeneous option {8.6}.

SUBROUTINE SUMS: takes 1 row of transition probability matrix and sums the results first by m_j's and then by j_{sum}'s. This assumes that the channel quantum numbers are appropriate for diatom-diatom scattering.

SUBROUTINE TAUMTS: calculates TAUE matrix {8.7} and perhaps its inverse.

SUBROUTINE TDUMP: writes TAUE {8.7} and perhaps its inverse to FORTRAN unit 8 for type-1 {8.9} second-case runs.

SUBROUTINE TITLES: prints description of sector-by-sector output or prints the sector-by-sector output itself.

SUBROUTINE TLOAD: reads TAUE {8.7} and perhaps its inverse from FORTRAN unit 8 for type-1 {8.9} second-case runs.

16.2 UTILITY ROUTINES

16.2.1 UTILITY ROUTINES FOR THE CRAY-2

The module "sdrcray.f" contains the following subroutines:

SUBROUTINE DATTIM: a utility subroutine which is called by the main program to return the calendar date and time. It calls a Cray system routines DATE(CDATE) and CLOCK(CTIME) where CDATE and CTIME are each CHARACTER * 10 variables, to obtain the date and time, and returns a character string CHARACTER*80 TIMDAT containing the date and time.

SUBROUTINE SECOND: a utility subroutine which is called by the main program and a number of other subroutines. It calls the Cray system subroutine to calculate CPU seconds taken by a task.

16.2.2 UTILITY ROUTINES FOR THE IBM RS/6000

The module "sdrrs6000.f" contains the following subroutines:

SUBROUTINE SECOND: a utility subroutine which is called by the main program and a number of other subroutines. It calls an IBM system subroutine, MCLOCK, to calculate CPU seconds taken by a task.

SUBROUTINE DATTIM: a utility subroutine which is called by the main program to return the calendar date and time. It calls a UNIX C Shell routine (included with the program) to obtain the date and time, and returns a character string CHARACTER*80 TIMDAT containing the date and time. It should be noted that the C Shell routine, "dateclock.c", must be compiled with the c compiler. The shell scripts included with the program do this automatically.

16.2.3 UTILITY ROUTINES FOR THE SUN SPARCSTATION:

The module "sdrsun.f" contains the following subroutines:

SUBROUTINE SECOND: a utility subroutine which is called by the main program and a number of other subroutines.

SUBROUTINE DATTIM: a utility subroutine which is called by the main program to return the calendar date and time. It calls the Sun version of the routine FDATE to find the date and time, and returns a character string CHARACTER*80 TIMDAT containing the date and time.

16.2.4 UTILITY ROUTINES FOR THE DEC ALPHA:

The module "sdralpha.f" contains the following subroutines:

SUBROUTINE SECOND: a utility subroutine which is called by the main program and a number of other subroutines. It should be noted that this routine is not consistently reliable in the current version.

SUBROUTINE DATTIM: a utility subroutine which is called by the main program to return the calendar date and time. It calls the DEC version of the routine FDATE to find the date and time, and returns a character string CHARACTER*80 TIMDAT containing the date and time.

17. MATRIX UTILITIES

The matrix utilities used in RMPROP have been taken from other sources. For complete documentation, the user is referred to references 7-12 in Section 2. In the table that follows, we list all the subroutines from other sources used to perform matrix and vector operations. Note that different versions of the code (see Sections 6.2-6.8 for more information on the versions of RMPROP) use different subroutines as well as different sources for the same subroutines. In the table, F refers to FORTRAN source codes distributed with the RMPROP distribution package, S referes to the Cray SCILIB library, and IB refers to the IBM RS/6000 libblas.a library.

Routines not called from the RMPROP distribution package are given a "-", although they may be called by other called library routines.

Routine	Type	2.1.1	2.1.1i	2.1.1ce	2.1.1cs	2.1.1xe	2.1.1xs
MINV	-	-	-	S	S	-	-
MXMA	-	-	-	S	S	-	-
EVDRSP	-	F	-	-	-	-	-
EVIRSP	-	-	F	-	-	-	-
EVCRSP	-	-	-	F	-	F	-
RS	-	-	-	-	S	-	S
D/SDOT	BLAS Level 1	F	IB	S	S	S	S
D/SNMR2	BLAS Level 1	F	IB	S	S	S	S
D/SASUM	BLAS Level 1	F	IB	S	S	S	S
D/SAXPY	BLAS Level 1	F	IB	S	S	S	S
D/SSCAL	BLAS Level 1	F	IB	S	S	S	S
D/SCOPY	BLAS Level 1	F	IB	S	S	S	S
D/SSWAP	BLAS Level 1	F	IB	-	-	-	-
ID/SAMAX	BLAS Level 1	F	IB	-	-	-	-
D/STRMV	BLAS Level 2	F	IB	-	-	-	-
DGER	BLAS Level 2	F	IB	-	-	-	-
D/SGEMV	BLAS Level 2	F	IB	-	-	-	-
D/SGEMM	BLAS Level 3	F	IB	-	-	S	S
D/STRSM	BLAS Level 3	F	IB	-	-	-	-
D/STRMM	BLAS Level 3	F	IB	-	-	-	-
D/STRTRI	BLAS Level 3	F	IB	-	-	-	-
D/SGETRF	LAPACK	F	F	-	-	S	S
D/SGETRS	LAPACK	F	F	-	-	S	S
D/SGETRI	LAPACK	F	F	-	-	S	S
D/SGETF2	LAPACK A	F	F	-	-	-	-
D/SLASWP	LAPACK A	F	F	-	-	-	-
D/STRTI2	LAPACK A	F	F	-	-	-	-
ILAENV	LAPACK A	F	F	-	-	-	-
LSAME	LAPACK A	F	F	-	-	-	-
XERBLA	LAPACK A	F	F	-	-	-	-

17.1 TIMINGS OF MATRIX UTILITY KERNELS

In order to decide on the optimal set of matrix utilities for RMPROP-version 2.0, it was necessary to compare the performance of various utilities on the Cray-2, Cray X-MP, and IBM RS/6000. The matrix operations for Version 2.1.2 have remained the same as in Version 2.0, so we will leave the following discussion intact for reference (NOTE: the Cray C90 is most similar in architecture to the Cray X-MP, although it is about four times as fast as a

Cray-2. These tests were performed before the Cray C90 was available.) The matrix operations which are required by RMPROP-Version 2.1.2 are matrix-matrix multiplication, solution of a real linear system of equations, inversion of a real matrix, and the determination of the eigensystem of a real symmetric matrix. We discuss the subroutines tested for each of the computational tasks listed above in turn. (Since the inversion of a real matrix is only performed during the asymptotic analysis at the end of a run by RMPROP, and the majority of the work in finding the inverse of a matrix is performed by one of the subroutines used to solve a system of linear equations, we have not performed timings of the inversion routines.)

The Cray-2 and Cray X-MP (and Cray C90) have 64-bit architectures, so that data which is single precision on the Cray machines are double precision on the other machines (see Section 6.3 above). This is reflected in the names of the utilities: subroutine DGEMM is called SGEMM on the Cray machines. However, for simplicity we refer to the routines by their double-precision names below.

For matrix-matrix multiplication, we have compared the two utilities MXMA, taken from the Cray SCILIB scientific subroutine library, and DGEMM, a Level-3 BLAS subroutine from the SCILIB subroutine library. Both MXMA and DGEMM exist as ANSI-standard FORTRAN 77 files, and as library subroutines on the Cray-2.

For the solution of a real linear solution of equations, the utilities we have compared include LUSOLV, which is a FORTRAN routine with unrolled DO LOOPs, SUBROUTINE LUBLAS, which is a version of SUBROUTINE LUSOLV with calls to the BLAS routines in place of the DO LOOPs, SUBROUTINE DGEFA and SUBROUTINE DGESL from the LINPACK linear algebra library, SUBROUTINE DGETRF and SUBROUTINE DGETRS from the LAPACK subroutine library, and SUBROUTINE MINV, from the Cray SCILIB scientific subroutine library. Of these routines, all but SUBROUTINE MINV exist as ANSI-standard FORTRAN 77 files. All of the routines except LUBLAS exist as library subroutines on the Cray-2 (again where 64 bit precision is SINGLE PRECISION on the Cray-2), and SUBROUTINE DGEFA and DGESL exist as library routines on the IBM RS/6000. SUBROUTINE LUBLAS, the LINPACK routines, and the LAPACK routines all contain calls to the BLAS, which exist as both ANSI-standard FORTRAN 77 files and library routines. We have compared FORTRAN versions of each of the routines where available, with both FORTRAN and library versions of the BLAS, and library versions of these routines as well.

For the determination of the eigensystem of a real symmetric matrix, we have compared SUBROUTINE RS, from the EISPACK scientific subroutine library, and SUBROUTINE EVVRSP, written by Stephen Elbert of Iowa State University. SUBROUTINE RS exists both as an ANSI-standard FORTRAN file and as an element of the Cray SCILIB library. SUBROUTINE EVVRSP contains calls to the BLAS, and we compare timings for various versions of EVVRSP with calls to both FORTRAN and library versions of the BLAS on both machines. We also consider versions of EVVRSP where the BLAS routines were replaced by FORTRAN DO LOOPs, and where FORTRAN elements of the subroutines ELAU and FREDA were replaced by calls to the BLAS.

Below we present comparisons of timings for each of the kernels above, for matrices of dimension 47 by 47, 236 by 236, and 1058 by 1058, taken from actual runs of RMPROP. The time given is the average of 10 executions of each routine, except for the 1058 by 1058 matrix, which is the average of 2 executions. The actual version of each kernel selected for the final program is described below the comparisons.

Timings of Matrix Multiplication (seconds)

	Cray-2			Cray X-MP			IBM RS/6000		
size of matrix -->	47	236	1058	47	236	1058	47	236	1058
FORTRAN MXMA	0.0046	0.41	29.5	0.0032	0.29	26.5	0.011	1.51	141.8
Cray SCILIB MXMA	0.0022	0.071	6.0	0.0036	0.14	11.4	N/A	N/A	N/A
FORTRAN DGEMM	0.0023	0.068	23.3	0.0037	0.30	26.0	0.012	1.52	142.8
Library DGEMM	0.0098	0.076	6.6	0.0014	0.13	11.9	0.0034	0.36	32.5

Timings of Linear Equation Solutions

	Cray-2			Cray X-MP			IBM RS/6000		
size of matrix -->	47	236	1058	47	236	1058	47	236	1058
LUSOLV (Fortran BLAS)	0.0088	1.2	58.1	0.082	0.44	33.8	0.010	1.39	243.5
LUBLAS (Fortran BLAS)	0.037	2.1	146.9	0.031	1.16	59.3	0.022	1.93	154.7
LUBLAS (Library BLAS)	0.016	0.67	36.1	0.015	0.64	44.3	0.22	1.91	150.7
LINPACK (FORTRAN)	0.023	2.2	109.2	0.019	0.89	50.3	0.019	2.76	2707.0
LINPACK (Library)	0.014	1.73	87.1	0.012	0.75	48.3	0.019	2.83	2747.6
LAPACK (FORTRAN, with FORTRAN BLAS)	0.017	0.87	50.2	0.133	0.66	48.2	0.022	2.43	212.0
LAPACK (FORTRAN, with Library BLAS)	0.0045	0.27	10.5	0.0045	0.24	16.0	0.0078	0.54	43.9
LAPACK (SCILIB Library)	0.0045	0.30	11.5	0.0042	0.26	16.8	N/A	N/A	N/A
MINV (SCILIB Library)	0.0017	0.12	9.2	0.0047	0.64	62.8	N/A	N/A	N/A

Timings of Eigensystem Analysis of Real Symmetric Matrix (seconds)

	Cray-2			Cray X-MP			IBM RS/6000		
size of matrix -->	47	236	1058	47	236	1058	47	236	1058
LAPACK DSYEV (FORTRAN, with FORTRAN BLAS)	0.300	1.94	103.2	N/A	N/A	N/A	0.480	47.2	4132.7
LAPACK DSYEV (FORTRAN, with Library BLAS)	N/A	N/A	N/A	N/A	N/A	N/A	0.350	33.3	2922.2
EVVRSP (FORTRAN, with FORTRAN BLAS)	0.055	2.02	100.4	0.049	1.49	71.3	0.035	2.13	164.1
EVVRSP (FORTRAN, with Library BLAS)	0.030	1.02	47.8	0.030	0.99	49.7	0.025	2.14	157.9
EVVRSP (FORTRAN, DO LOOPs replace BLAS)	0.028	1.05	47.8	0.025	0.88	49.4	0.030	2.58	207.4
EVVRSP (FORTRAN, with FORTRAN BLAS in ELAU and FREDA)	0.08	2.98	152.2	0.071	2.10	102.0	0.040	2.34	163.7
EVVRSP (FORTRAN, with Library BLAS in ELAU and FREDA)	0.035	1.16	48.4	0.035	1.29	62.3	0.040	2.20	158.7
RS (FORTRAN)	0.031	2.09	132.6	0.036	1.55	134.7	0.448	53.4	6122.9
RS (SCILIB)	0.029	1.42	102.1	0.025	1.28	98.7	N/A	N/A	N/A

Examination of the tables above reveals the following trends. For matrix-matrix multiplication, the FORTRAN version of DGEMM is superior to the FORTRAN version of MXMA on both the Cray and the IBM RS/6000. The library version of MXMA is slightly faster than the library version of DGEMM on the Cray, and the library version of DGEMM is vastly superior to either the FORTRAN version of DGEMM or MXMA on the IBM RS/6000. Therefore, for matrix-matrix multiplication, version 2.1.2 uses the FORTRAN version of DGEMM, versions 2.1.2cs and 2.1.2ce use the SCILIB version of SUBROUTINE MXMA, and versions 2.1.2xs, 2.1.2xe, and 2.1.2i use the machine-appropriate library versions of DGEMM.

For the solution of systems of linear equations, the fastest FORTRAN code is taken from LAPACK, on both the Cray-2 and the IBM RS/6000. However, when library modules are included, the fastest routine on the Cray-2 is SUBROUTINE MINV, from the Cray SCILIB library, while the fastest routine on the IBM RS/6000 is taken from LAPACK, including calls to the IBM library BLAS. The portable version 2.1.2 uses the FORTRAN LAPACK routines, while the library LAPACK routines are used in versions 2.1.2i, 2.1.2xs, and 2.1.2xe. The SCILIB version of MINV is used in versions 2.1.2cs and 2.1.2ce.

For the solution of the eigensystem of a real symmetric analysis, the choice of the optimal subroutine is more complex. Among the FORTRAN modules, the fastest subroutine on the Cray-2 is a version of Stephen Elbert’s EVVRSP where all of the BLAS routines are replaced by equivalent FORTRAN DO LOOPs (this version is called EVCRSP). The fastest version on the IBM RS/6000 is a version of Stephen Elbert’s EVVRSP where elements of the subroutines ELAU and FREDA are replaced by calls to the FORTRAN BLAS (this version is called EVIRSP). When library modules are included, the fastest subroutine on the Cray-2 varies with the size of the matrix concerned. SUBROUTINE RS from the SCILIB routine is the fastest for matrices of size 236 by 236 or smaller, but EVCRSP with calls to the library BLAS is the fastest routine for the largest matrix tested. On the IBM RS/6000, the fastest routine which includes any library routines is EVIRSP with calls to the library BLAS.

In order to make the version 2.1.2 as fast as possible, we have selected from the routines described above to perform matrix manipulations within RMPROP-Version 2.1.2. In summary, these matrix routines include three routines for the solution of systems of linear equations (DGETRF and DGETRS/DGETRI), a routine for matrix-matrix multiplication (DGEMM), and a routine for the eigensystem analysis of a real symmetric matrix (EV2RSP). Version 2.1.2cs, 2.1.2ce, 2.1.2cx, 2.1.2ce, and 2.1.2i include specialized versions of these routines intended to improve the program’s performance.

In addition, version 2.1.2ce and 2.1.2cx of the program allows calls to the Cray Research Inc. scientific library (SCILIB) routines for the eigensystem analysis of a real matrix, SUBROUTINE RS, for matrix-matrix multiplication, MXMA, and for the solution of a linear system of equations, SUBROUTINE MINV, and version 2.1.2i also includes subroutine calls to library versions of the BLAS subroutines to speed up execution.

We first discuss the subroutines above (DGETRF, DGETRS, DGETRI, DGEMM, and EV2RSP) which are common to versions 2.1.2, 2.1.2xs, 2.1.2xe, and 2.1.2i of the program, and then discuss those routines from the LAPACK and the BLAS libraries which are called by these matrix utilities used in the distribution version of the program, whether or not these subsidiary routines are concerned with matrix manipulations. Next, we supply the names of the routines which are unique to

version 2.1.2cx and 2.1.2ce of the program, their subsidiary subroutines where applicable, and the source from which we have obtained them. Finally, we do the same for the IBM RS/6000 version of the program.

18. FATAL ERROR MESSAGES

The program attempts to detect run time errors and input errors whenever possible. If an error is detected, the program ceases execution with an error message of the form STOP N, where N is an integer, and it writes this message to FORTRAN unit 6. The list below gives the messages by number for each routine in which they occur.

In program RMPROP:

'STOP 5099' Is not an error message but is issued for runs with LJRI equal to .TRUE. when reading and echoing of FORTRAN unit 5 is complete.

'STOP 5100' NDIM {8.2} has exceeded the dimension ID.

'STOP 5200' The number of type-2 second energies {8.9} exceeded the dimension NNE.

'STOP 5300' Is a flag when the user tries to run with both L(I) and OAM(I) not equal to zero. The L(I) should be used for general potentials only, and the OAM(I) for spherically symmetric potentials only.

'STOP 5400' Occurs when the left-hand side of the first sector would have been less that zero.

'STOP 5500' Is a flag when the user tries to run more than one set of stepsize criteria and more than one energy at the same time.

'STOP 5600' Occurs when, for type-2 {8.9} second-case runs, the left hand sides of the first sector for each energy are not lined up with one another.

'STOP 5700' Occurs when there are problems with diagonalizing the D matrix in the asymptotic region before beginning the propagation.

'STOP 5800' When there is an error in the diagonalization of the D matrix at the first sector.

'STOP 5900' Occurs when reading in restart information from FORTRAN unit 17 or FORTRAN unit 18 and the variable 'LTHREW' is true. This error message is also triggered when the program tries to read data from a non-existent FORTRAN unit 17 or FORTRAN unit 18. LTHREW is set to true by the main program when convergence with respect to the number of sectors has been reached, or RMAX has been exceeded {8.4,8.15}.

'STOP 6000' Is not an error, but is triggered when the program is finished writing to FORTRAN unit IUNXX: writing to FORTRAN unit IUNXX is an infrequently used option and occurs when the sector number, IS, equals the input value of ISAVV.

'STOP 6100' Is the error message whenever there is a problem diagonalizing the D matrix at finite distance.

'STOP 6200' Is the flag if in program rmprop the symmetry check of off-diagonal elements of R is failed.
4

'STOP 6300' Is the message for normal, successful completion for single energy, single basis set, single stepsize criterion runs.

'STOP 6400' Occurs when the number of sectors exceeds the dimension NSECT before convergence for single energy runs.

'STOP 6500' Occurs when the number of sectors exceeds the dimension NSECT before convergence for second-case runs.

In SUBROUTINE GNSCAT:

'STOP 981' Refers to testing the channel ordering at intermediate distances or during the final asymptotic analysis. If a closed channel occurs during the ordering of the open channels or if an open channel is among the closed channels the code stops with this error message.

In SUBROUTINE RPROP:

'STOP 13' Occurs in SUBROUTINE RPROP when a zero eigenvalue is encountered.

'STOP 133' Occurs in SUBROUTINE RPROP when the average value of the relative error of off-diagonal elements of the R matrix is greater than
4
or equal to the tolerance set by the program.

In SUBROUTINE TAUMTS:

'STOP 111' Occurs when there is an error in the inversion of the TAUE matrix.

19. TEST RUNS AND COMPILATION

This section of the manual contains some sample input data sets for RMPROP and copies of selected subsets of the output data. The systems involved in the examples are:

HF-HF R-R,T transitions
(see references 5, 9, and 10 in Section 21)

He-HF R-R,T transitions
(see references 6 and 7 in Section 21)

$He-H_2$ collinear V-V,T transitions
(see reference 8 in Section 21)

General atom-rigid rotor diatom R-R,T transitions
(see reference 11 in Section 21 and the Appendix of the METECC-94 Chapter)

Electron-H atom scattering
(see reference 12 in Section 21)

$He-I_2$ three-dimensional R-R,T scattering
(see references 13 and 14 in Section 21)

$He-H_2$ three-dimension V,R-T scattering
(see references A, B, and C in Section 21)

Various options were used for each of these systems to test many different available options. The HF-HF runs use several different potentials - the Schwenke-Truhlar potential energy surface (reference 5 in Section 21), the Poulsen-Billing-Steinfeld surface (reference 9 in Section 21), and the surface of Alexander and DePristo (reference 10 in Section 21).

The He-HF runs use the potential of Collins and Lane (reference 6 in Section 21), and the potential for the $He-H_2$ system, for the collinear approach of He to H_2 is from reference 8 of Section 21.

The generalized atom-rigid rotor diatomic scattering runs follow the treatment of Lester and Bernstein (reference 11 of Section 21). This test run is of particular interest since the coupled equations were originally solved by a different method (de Vogelaere's algorithm) and using a different set of units. Lester and Bernstein used reduced (dimensionless) units, which were converted to atomic units for this program.

The $He-I_2$ rotational scattering illustrates the expansion of the diatomic potential of Schwenke and Truhlar (reference 13 in Section 21) in Legendre functions, in order that the angular integrals in the potential matrix elements [A12] become analytic.

We adopt the following names for test runs:

hfhfstr1, hfhfstr2, hfhfstr3, hfhfstr4, hfhfstr5
hehfr1
heh2cdr1, heh2cdr2
hfhfpbsr1
hfhfadr1, hfhfadr2
atdir1
ehydr1
hei2r1
heh2tkr1, heh2tkr2

The convention followed is that the first group of 4 letters refers to the system under study, for example hfhf or heh2. If there is more than one potential surface being used for a given system, the initials of the authors of the potential surface are given after the name of the system, e. g. "hfhfpbs" is the Poulsen-Billing-Steinfeld surface for HF-HF but "hehf" has no author's initials following since we only study one surface for He-HF. Finally the r1, r2, etc. suffix refers to the number of the run for a given system and potential surface. Therefore by this convention "hfhfstr2" is the second test run on the HFHF system on the ST surface.

A convenient C Shell script is provided for running the test runs. To use this script, called runtest.csh. In addition the file runtest.csh should be executable (if it is not executable one may make it executable with the command chmod 750 runtest.csh).

All sixteen test runs have been tested on a Cray-2, a Cray C90, and an IBM RS/6000. In order to maintain portability, version 2.1.2 of the code was run in all cases. This differs from RMPROP-Version 1.0, where the Cray-2 ran the Cray UNICOS version for all elements of the test suite except hfhfstr2 and hfhfstr3, which ran the distribution version.

The compilation of the runs in the test suite is performed separately for each run in the test suite by the UNIX make utility. The make utility is a general purpose utility which may be used to update and to maintain files within a UNIX directory. By default, the make utility assumes that files ending in ".f" are FORTRAN modules and that files ending in ".o" are object modules. When the make utility is invoked by the C Shell script, the date of creation of the executable image for that test run is compared to the latest modification date of each of the FORTRAN subprograms which, when compiled and linked, comprise that image. Compilation of an individual subprogram to create a new object module and linkage of the new module with existing object modules to create a new executable image is performed only if the last modification date for that subprogram is later than that of the executable image, or if the object module corresponding to the FORTRAN subprogram is missing from the current working directory.

The file which we use to access the make utility is called "Makefile.mak". This file contains a number of UNIX environment variable assignments. These environment variables are defined either by the C Shell script which drives the specific test run (hereafter referred to as {TEST}.csh) or by runtest.csh. The file runtest.csh chooses the environment variables which vary by machine and operating system (UNICOS for the Cray, and AIX for the IBM RS/6000) and sets variables which denote the directories files are

located for the benefit of {TEST}.csh. The files {TEST}.csh for individual test runs set environment variables specific to the test run. The file "Makefile.mak" uses the following environment variables:

Variable	Set by	Meaning
FTN	runtest.csh	command for FORTRAN compiler step
LOAD	runtest.csh	command for FORTRAN linking step
FFLAGS	runtest.csh	options to FORTRAN compiler step
SDR	runtest.csh	machine-dependent source code
PSSC	{TEST}.csh	potential-surface source code
CMD	{TEST}.csh	executable name

NOTE: All of the {TEST}.csh files can be modified simply to run without the use of runtest.csh. Comments within each of these scripts aid the user in modifying these to run individually. The file {TEST}.csh should be moved from directory script to the top directory of the distribution package. Then the script must be altered to set the environement variables

FTN	(see above)
LOAD	(see above)
FFLAGS	(see above)
DIR	top directory of distribution package
TDIR	directory test (or where input files are kept)
FDIR	directory src (or where source code is kept)
SDIR	directory script (or where Makefile.mak is kept)

The file "Makefile.mak" assumes that any FORTRAN subprograms other than those of the elements of RMPROP-Version 2.1.2 (rmprop2.f, gnscat2.f, and exscat2.f) or machine-dependent routines (sdrcray.f, sdrrs6000.f, etc) will be defined by the variable PSSC. Therefore, if the user wishes to use the supplied C Shell scripts and Makefile.mak to run this program with his or her own potential routines, then he or she must define PSSC in the new script before calling Makefile.mak.

In order to compile and link the programs listed in the test suite, ensure that "runtest.csh" is in the top directory of the distribution package. All other files should be in the directories specified in Sections 5 and 20. Enter the command

```
runtest.csh TESTNAME
```

where TESTNAME is a name of an element of the test suite, such as hfhfstr1, hehfr1, etc. The compilation and linking are performed automatically when the make utility is called by the C Shell script, and as explained above, will only need to be repeated if one of the FORTRAN modules is modified or an object file deleted. CAUTION: a change to the PARAMETER statement in the file parameter.inc or xxx.param (see Section 15 of this manual) will necessitate removal of existing object files (those with a suffix of '.o') and re-compilation of the program. The rest of this section gives detailed INPUT and OUTPUT for the elements of the test suite.

19.1 TEST RUN hfhfstr1

HF(v = 0,j = 0) + HF(v = 0,j = 0) -> HF(v = 0,j = j') + HF(v = 0,j = j')

This sample run solves the close coupling equations for fully 3-dimensional scattering of a vibrating rotor from another vibrating rotor at non-zero total angular momentum. The run is performed at a single energy, with 47 channels, and uses the "fast" propagator, NPROP = 2 {8.10}.

The run uses fixed stepsizes throughout the propagation, and it saves restart information alternating between FORTRAN units 17 and 18. The asymptotic analysis is performed at 2 intermediate distances, and comparison is made of the phases and magnitudes of the scattering matrix elements coupling the first channel to the second, third, and fourth channels, using the input tolerances EPSMAG = 0.001 and EPSPH = 0.001 {8.15} respectively. The asymptotic boundary conditions are enforced by matching to Bessel functions using the asymptotic (r = 5000.0 bohr) wave vectors.

In order to run this test program, perform the following steps:

1) Ensure that the following files are all present:

test/hfhfstr1.4	(input data for FORTRAN unit 4)
test/hfhfstr1.5	(input data for FORTRAN unit 5)
test/hfhfstr1.param	(include file of parameters copied to parameter.inc)
runtest.csh	(C Shell script to run program)
script/hfhfstr1.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)

2) Enter the command

```
runtest.csh hfhfstr1
```

3) Output files generated

hfhfstr1.out	standard output (UNIT 6)
hfhfstr1.3	date and time of the run (UNIT 3)
hfhfstr1.15	short output (UNIT 15)

It should be noted that, even though the input file (FORTRAN unit 5) calls for the asymptotic analysis to be performed at two intermediate distances (380.0 and 390.0 bohr, respectively) as well as at RMAX (400.0 bohr), FORTRAN unit 15 only contains the scattering matrix at 390.0 bohr. This is because IOPT = 3 so that the scattering matrix is only printed at the "last" sector. In the present run, the last sector is at 390.0 because when the program compared the magnitude and the phase of the S-matrix elements between the initial and final states given in the input arrays IND(NIND,1) and IND(NIND,2), at the two intermediate distances, (380.0 bohr and 390.0 bohr), all of the S-matrix elements agreed within the tolerances set by the input values EPSMAG and EPSPH, and so the program ceased execution instead of continuing the propagation to 400.0 bohr. The use of these convergence tests is helpful but can lead to unwarranted indications of convergence if used in an inappropriate fashion (see hfhfstr3 and {8.15} above).

The output to FORTRAN unit 6 also includes timing information on the various routines and a listing of the center of each sector and the stepsize at each sector: FORTRAN unit 6 is the only other file with user-readable output created by this run.

Input to FORTRAN unit 4 (for potential routine):

```
1.733
    1
   10
    1
T
    3
TEST RUN FOR ROTATIONAL-ROTATIONAL TRANSFER AT 76 meV.
SCHWENKE-TRUHLAR INTERACTION POTENTIAL PLUS VIBRATIONAL DEPENDENCE.
JSUM =4
    1
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0
    0    1    1    1    1    1    1    2    2    2
    2    2    2    2    2    2    2    2    3    3
    3    3    2    2    2    2    2    2    2    2
    2    3    3    3    3    3    3    3    3    3
    3    3    4    4    4    4    4
    0    0    0    1    1    1    1    0    0    0
    1    1    1    1    1    1    1    1    0    0
    0    0    2    2    2    2    2    2    2    2
    2    1    1    1    1    1    1    1    1    1
    1    1    0    0    0    0    0
    0    1    1    0    2    2    2    2    2    2
    1    1    2    2    3    3    3    3    3    3
    3    3    0    2    2    2    4    4    4    4
    4    2    2    2    3    3    3    4    4    4
    4    4    4    4    4    4    4
    0    0    0    0
   23    4
    0    0    1    0    2    1    1    0    3    1
    2    1    2    3    1    2    2    2    2    3
    2    3    3
    0    1    0    2    0    1    1    3    0    2
    1    2    1    1    3    2    2    2    3    2
    3    2    3
    0    0    0    0    0    0    1    0    0    0
    0    1    1    0    0    0    1    2    0    0
    2    2    0
1744.604       0.01889699     1.733          0.0            T 1000.0            1
   60   40   10    0    3    1
    7    4    7   12    0
   10
    0
```

Input to FORTRAN unit 5 (for RMPROP):

```
SCHWENKE-TRUHLAR SURFACE HF-HF J=100 AND E = 76 meV
TEST SUITE RUN 1
FT
   47 4000   13    1    2   10    1
    0  300    0   30    0
1.823454D+4       47  100
  100   99  101  100   98  100  102   98  100  102
   99  101   99  101   97   99  101  103   97   99
  101  103  100   98  100  102   96   98  100  102
  104   98  100  102   98  100  102   96   98  100
  102  104   96   98  100  102  104
    1   -1    0
   47
0.021444753            4.00       0
        0.1200        0.1200        400.0        h1,h2,rmax
    1
         1.0D-1
         0.0129
         1.2D-2
         0.1200
         0.1D-5
         0.1D-5
         0.1200
         3.4D+3
         3.4D+3
         4.0D+2
         9.0D+3
    0
 0.0
         1.D-03       400.0       2               epsred,rbig,nprop
FTFFFFFFFF
TTFFFFFFFT
         1.D-03     5000.0                        epsdr,rasye
TTF    0    2    0
    2
380.0          390.0
TTFFTTFTFFFFFFFF
TTFFTTFTFFFFFFFF
   3   3
 1.D-3          1.D-3
   3
   1   1   1
   2   3   4
```

19.2 TEST RUN hfhfstr2

HF(v = 0,j = 0) + HF(v = 0,j = 0) -> HF(v = 0,j = j') + HF(v = 0,j = j')

This test run differs from the previous test run ONLY in that the slow form of the propagator (NPROP = 0) is used, to compare the timing.

In order to run this test program, perform the following steps:

1) Ensure that the following files are available:

test/hfhfstr2.4	(input data for FORTRAN unit 4)
test/hfhfstr2.5	(input data for FORTRAN unit 5)
test/hfhfstr2.param	(include file of parameters copied to parameter.inc)
runtest.csh	(C Shell script to run program)
script/hfhfstr2.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)

2) Enter the command

```
runtest.csh hfhfstr2
```

3) Output files generated

hfhfstr2.out	standard output (UNIT 6)
hfhfstr2.2	angular integrals (saved for restart in hfhfstr3)
hfhfstr2.3	date and time of the run (UNIT 3)
hfhfstr2.15	short output (UNIT 15)
hfhfstr2.17	restart information (UNIT 17, nonASCII)
hfhfstr2.18	restart information (UNIT 18, nonASCII)

As can be seen from the discussion of timings in Section 4 of this manual, the NPROP = 2 option (using the distribution version) resulted in a speedup of 25% on the Cray-2, and 50% on the IBM RS/6000, while giving the same final answers.

Input to FORTRAN unit 4 (for potential routine):

Identical to that for hfhfstr1

Input to FORTRAN unit 5 (for RMPROP 2.1.2):

```
SCHWENKE-TRUHLAR SURFACE HF-HF J=100 AND E = 76 meV
TEST SUITE RUN 2
FT
   47 4000   13    1    2   10    1
    0  300    0   30    0
1.823454E+4       47  100
  100   99  101  100   98  100  102   98  100  102
   99  101   99  101   97   99  101  103   97   99
  101  103  100   98  100  102   96   98  100  102
  104   98  100  102   98  100  102   96   98  100
  102  104   96   98  100  102  104
    1   -1    0
   47
0.021444753            4.00       0
        0.1200        0.1200        400.0
    1
         1.0E-1
         0.0129
         1.2E-2
         0.1200
         0.1E-5
         0.1E-5
         0.1200
         3.4E+3
         3.4E+3
         4.0E+2
         9.0E+3
    0
 0.0
         1.E-03      400.0        0
FTFFFFFFFF
TTFFFFFFFT
         1.E-03     5000.0
TTF    0    2    0
    2
380.0          390.0
TTFFTTFTFFFFFFFF
TTFFTTFTFFFFFFFF
   3   3   3
 1.E-3          1.E-3
   3
   1   1   1
   2   3   4
```

19.3 TEST RUN hfhfstr3

HF(v = 0,j = 0) + HF(v = 0,j = 0) -> HF(v = 0,j = j') + HF(v = 0,j = j')

This run continues the propagation begun by hfhfstr2 and shows that the program runs correctly when reading restart information. This run reads restart information from FORTRAN unit 17 (written by hfhfstr2) and uses NPROP = 2. The asymptotic boundary conditions are again imposed by matching to Ricatti-Bessel functions using the asymptotic wave vectors. The asymptotic analysis is performed at one intermediate distance, 390.0 bohr, and rmax, 400.0 bohr, so the elements of the scattering matrix written to hfhfstr3.15 differ slightly from those of hfhfstr2.15. This test run also reads the angular integrals written to UNIT 2 by hfhfstr2 instead of recalculating them.

In order to run this test program, perform the following steps:

1) Ensure that the following files are available:

hfhfstr2.3	(time and date of previous run)
hfhfstr2.2	(integral file from previous run)
hfhfstr2.17	(restart information from previous run)
test/hfhfstr3.4	(input data for FORTRAN unit 4)
test/hfhfstr3.5	(input data for FORTRAN unit 5)
test/hfhfstr3.param	(include file of parameters copied to parameter.inc)
runtest.csh	(C Shell script to run program)
script/hfhfstr3.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)

2) Enter the command

```
runtest.csh hfhfstr3
```

3) Output files generated:

hfhfstr3.out	standard output (UNIT 6)
hfhfstr3.3	date and time of the run (UNIT 3)
hfhfstr3.15	short output (UNIT 15)

Investigation of the output of FORTRAN unit 6 for hfhfstr3 shows that the scattering matrix element linking channel 2 to channel 2 is not converged to the input tolerance, 0.001 relative error, when comparing the results at r = 390.0 bohr to those at r = 400.0 bohr. The program then stops execution because the distance is greater than the input rmax. This occurrence suggests that, even though the results of the first test run indicated that the results of the calculation at r = 380.0 bohr agreed with those at r = 390.0 bohr, in reality, not all of the scattering matrix elements are converged to the same tolerance. This illustrates the need for caution when allowing the program to converge the scattering matrix elements: intelligent direction of the program on the part of the user is still required.

Input to FORTRAN unit 4 (for potential routine):

Identical to that for hfhfstr1

Input to FORTRAN unit 5 (for RMPROP 2.1.2):

```
SCHWENKE-TRUHLAR SURFACE HF-HF J=100 AND E = 76 meV
TEST SUITE RUN 3
FT
   47 4000   13    1    2   10    1
   17  300    0   30    0
1.823454E+4       47  100
  100   99  101  100   98  100  102   98  100  102
   99  101   99  101   97   99  101  103   97   99
  101  103  100   98  100  102   96   98  100  102
  104   98  100  102   98  100  102   96   98  100
  102  104   96   98  100  102  104
    1   -1    0
   47
0.021444753            4.00       0
        0.1200        0.1200        400.0
    1
         1.0E-1
         0.0129
         1.2E-2
         0.1200
         0.1E-5
         0.1E-5
         0.1200
         3.4E+3
         3.4E+3
         4.0E+2
         9.0E+3
    0
 0.0
         1.E-03      400.0        0
FTFFFFFFFF
TTFFFFFFFT
         1.E-03     5000.0
TTF    0    2    0
   0
380.0
TTFFTTFTFFFFFFFF
TTFFTTFTFFFFFFFF
   3   3   3
 1.E-3           1.E-3
   3
   1   1   1
   2   3   4
```

19.4 TEST RUN hfhfstr4

HF(v = 0,j = 0) + HF(v = 0,j = 0) -> HF(v = 0,j = j') + HF(v = 0,j = j')

The purpose of this test run is to give a run for the Schwenke-Truhlar surface which is not so time consuming as the other test runs for this potential. All of the parameters have the same values as those for hfhfstr1, except that the total angular momentum is 50 instead of 100, and ID is only 10 instead of 47.

In order to run this test program, perform the following steps:

1) Ensure that the following files are available:

test/hfhfstr4.4	(input data for FORTRAN unit 4)
test/hfhfstr4.5	(input data for FORTRAN unit 5)
test/hfhfstr4.param	(include file of parameters copied to parameter.inc)
runtest.csh	(C Shell script to run program)
script/hfhfstr4.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)

2) Enter the command

```
runtest.csh hfhfstr4
```

3) Output files generated

hfhfstr4.out	standard output (UNIT 6)
hfhfstr4.3	date and time of the run (UNIT 3)
hfhfstr4.15	short output (UNIT 15)

Input to FORTRAN unit 5 (for RMPROP 2.1.2):

```
SCHWENKE-TRUHLAR SURFACE HF-HF J= 50 AND E = 76 meV
TEST SUITE RUN 4
FT
   47 4000   13    1    2   10    1
    0  300    0   30    0
1.823454D+4       10   50
   50   49   51   50   48   50   52   48   50   52
    1   -1    0
   10
0.021444753           4.00       0
        0.1200        0.1200        400.0
    1
         1.0D-1
         0.0129
         1.2D-2
         0.1200
         0.1D-5
         0.1D-5
         0.1200
         3.4D+3
         3.4D+3
         4.0D+2
         9.0D+3
    0
 0.0
         1.D-03       400.0       2
FTFFFFFFFF
TTFFFFFFFT
         1.D-03     5000.0
TTF    0    2    0
    2
380.0          390.0
TTFFTTFTFFFFFFFF
TTFFTTFTFFFFFFFF
   3   3
 1.D-3           1.D-3
   3
   1   1   1
   2   3   4
```

```
Input to FORTRAN unit 4 (for potential routine):

1.733
    1
   10
    1
T
    3
TEST RUN FOR ROTATIONAL-ROTATIONAL TRANSFER AT 76 meV.
SCHWENKE-TRUHLAR INTERACTION POTENTIAL PLUS VIBRATIONAL DEPENDENCE.
JSUM = 2
    1
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0
    0    1    1    1    1    1    1    2    2    2
    0    0    0    1    1    1    1    0    0    0
    0    1    1    0    2    2    2    2    2    2
    0    0    0    0
   23    4
    0    0    1    0    2    1    1    0    3    1
    2    1    2    3    1    2    2    2    2    3
    2    3    3
    0    1    0    2    0    1    1    3    0    2
    1    2    1    1    3    2    2    2    3    2
    3    2    3
    0    0    0    0    0    0    1    0    0    0
    0    1    1    0    0    0    1    2    0    0
    2    2    0
1744.604       0.01889699     1.733          0.0            T 1000.0            1
   60   40   10    0    3    1
    7    4    7   12    0
   10
    0
```

19.5 TEST RUN hfhfstr5

HF(v = 0,j = 0) + HF(v = 0,j = 0) -> HF(v = 0,j = j') + HF(v = 0,j = j')

The purpose of this test run is to demonstrate the multiple step-size region options of RMPROP. The calculation is identical to the one performed in hfhfstr1, but now the step-size is varied by the parameter EPSA, EPSB, and EPSC in three regions of the radial coordinate denoted by REPSA and REPSB. The transition probabilities greater than 10E-4 in the first column of the transition probability matrix are converged to 1% of those in hfhfstr1, but since the stepsize is permitted to vary, this testrun takes less than 1/3 the number of steps. This also means that the work for this run is 1/3 less than for hfhfstr1.

In order to run this test program, perform the following steps:

1) Ensure that the following files are available:

test/hfhfstr5.4	(input data for FORTRAN unit 4)
test/hfhfstr5.5	(input data for FORTRAN unit 5)
test/hfhfstr5.param	(include file of parameters copied to parameter.inc)
runtest.csh	(C Shell script to run program)
script/hfhfstr5.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)

2) Enter the command

```
runtest.csh hfhfstr5
```

3) Output files generated

hfhfstr5.out	standard output (UNIT 6)
hfhfstr5.3	date and time of the run (UNIT 3)
hfhfstr5.15	short output (UNIT 15)

Input to FORTRAN unit 5 (for RMPROP 2.1.2):

```
SCHWENKE-TRUHLAR SURFACE HF-HF J=100 AND E = 76 meV
TEST SUITE RUN 1
FT
   47 4000   13    1    2   10    1
    0  300    0   30    0
1.823454D+4       47  100
  100   99  101  100   98  100  102   98  100  102
   99  101   99  101   97   99  101  103   97   99
  101  103  100   98  100  102   96   98  100  102
  104   98  100  102   98  100  102   96   98  100
  102  104   96   98  100  102  104
    1   -1    0
   47
0.021444753            4.00       0
        0.0120        0.0120        400.0
    1
         8.0D-2    epsa
         0.4129    epsb
         9.2D-1    epsc
         0.0120    hmina
         0.0120    hminb
         0.0120    hminc
         0.2200    hmaxa
         3.4D+0    hmaxb
         9.4D+1    hmaxc
         5.0D+1    repsa
         8.0D+1    repsb
    0
 0.0
         1.D-03       400.0       2
FFFFFFFFF
TTFFFFFFT
         1.D-03     5000.0
TTF    0    2    0
    2
380.0          390.0
TTFFTTFTFFFFFFFF
TTFFTTFTFFFFFFFF
   3   3
 1.D-3           1.D-2
   6
   1   1   1   1   1   2
   2   3   4   5   6   3
```

Input to FORTRAN unit 4 (for potential routine):
Same as for hfhfstr1.

19.6 TEST RUN hehfr1

He + H-F (rigid rotor in state j) -> He + H-F (rigid rotor in state j')

This test run is for a much simpler physical system than the first four test runs: instead of a collision of two vibrating diatoms, it is the collision of a structureless particle with a non-vibrating diatom. This test run shows how the program can run multiple energies simultaneously, and it allows for the step-size to vary beginning at the third sector. The logical variable LPSYM(3) is turned on, which prints symmetry checks on the R4MT matrix at each energy and sector. The T matrix is written to FORTRAN unit 10, and the lower triangle symmetrized reactance matrix is written to FORTRAN unit 7. This test also illustrates two important features of the convergence tests on elements of the scattering matrix: first, that if closed channels are included in the indices of scattering matrix elements to be tested for convergence, they are ignored; and that if multiple energies are performed, then only scattering matrix elements at the first energy are tested for convergence with respect to RMAX. Finally, this test run shows that if asymptotic analyses are desired at more than one intermediate distance, they must be at distances sufficiently removed that they will not be performed in adjacent sectors. This is to help avoid the illusion of convergence when comparing scattering matrix elements at closely spaced separations. This test uses prevgr1.f and vgr1.f.

In order to run this test program, perform the following steps:

1) Ensure that the following files are available:

test/hehfr1.4	(input data for FORTRAN unit 4)
test/hehfr1.5	(input data for FORTRAN unit 5)
test/hehfr1.param	(include file of parameters copied to parameter.inc)
runtest.csh	(C Shell script to run program)
script/hehfr1.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)

2) Enter the command

```
runtest.csh hehfr1
```

3) Output files generated:

hehfr1.out	standard output (UNIT 6)
hehfr1.3	date and time of the run (UNIT 3)
hehfr1.7	symmetrized reactance matrix (UNIT 7)
hehfr1.10	transition matrix (UNIT 10)
hehfr1.15	short output (UNIT 15)

Input to FORTRAN unit 4 (for potential routine):

```
T
    0    1    2    2    3    3
    6    4  120    5
128.275   2.494     3.42        6
 10.356   2.151     6.78        7
 29.253   2.379     1.55        6
 10.739   2.173     1.41        7
  3.875   2.115     1.23        8
  6  5.0
```

Input to FORTRAN unit 5 (for RMPROP 2.1.2):

```
COLLINS, LANE COLLINEAR He-HF 2 ENERGIES
TEST SUITE RUN 5
FT
    6 4000    1    1    1    1    2
    0  100    0    0    0
0.612000E+4         6    4
    4    3    2    4    1    3
    1   -2    0
    6
0.00062472          3.50          0
0.0018374           3.50          0
0.0010         0.0010         26.58
    1
         0.0300
         0.0300
         0.0300
         0.1E-6
         0.1E-6
         0.1E-6
         0.5000
         0.5000
         0.5000
         0.1000
         1.0E+1
    0
 0.0
         1.E-03          20.0     2
TFFTFFFFFF
TTFFFFFFFT
         1.E-03        1000.0
TTF    0    2    2
    2
      24.0           24.2
TTTFFTFFTFFFTTFTFFF
TTTFFTFFTFFFTTFTFFF
   3   2   3
 1.E-3
   3
   3   4   5
   3   4   5
```

19.7 TEST RUN heh2cdr1

He + H-H (harmonic oscillator in state j)
-> He + H-H (harmonic oscillator in state j')

This test run is for the collinear collision of a structureless atom with a harmonic oscillator. It tests the multiple basis set option of the program. It should be noted that the files FORTRAN unit 8 and of FORTRAN unit 14 created by this test run are non-ASCII files and so are not presented in the write-up. In addition the option LPSINF is read in as .TRUE. so that a description of the sector output is written to standard output.

In order to run this test program, perform the following steps:

1) Ensure that the following files are available:

test/heh2cdr1.4	(input data for FORTRAN unit 4)
test/heh2cdr1.5	(input data for FORTRAN unit 5)
test/heh2cdr1.param	(include file of parameters copied to parameter.inc)
runtest.csh	(C Shell script to run program)
script/heh2cdr1.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)

2) Enter the command

```
runtest.csh heh2cdr1
```

3) Output files generated

heh2cdr1.out	standard output (UNIT 6)
heh2cdr1.3	date and time of the run (UNIT 3)
heh2cdr1.8	restart information (UNIT 8)
heh2cdr1.14	restart information (UNIT 14)
heh2cdr1.15	short output (UNIT 15)

Input to FORTRAN unit 5 (for RMPROP 2.1.2):

```
HELIUM-HARMONIC OSCILLATOR COLLISION 4 BASIS SET RUN
TEST SUITE RUN 6
FT
   15  800    1    1    2   10    1
    0  100    0    0    0
0.245080E+4       14    0
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0
    4   -1    0
   14   12   10    8
0.16016000      1.20              0
 0.0010        0.0010          6.50
    1
         0.1500
         0.1500
         0.1500
         0.1E-6
         0.1E-6
         0.1E-6
         3.0000
         3.0000
         3.0000
         0.1500
         1.0E+1
    0
 0.0
         1.E-30        5.0        2
FFFFFFTFFF
TTFFFFFFFT
         1.E-02     200.0
TFT    0    2     2
    0
   6.2
TTFFFTFFFFFFFFF
TTFFFTFFFFFFFFF
   3   3   3
 1.E-3           1.E-3
   3
   1   1   1
   2   3   4
```

Input to FORTRAN unit 4 (for potential routine):

```
TT
9.480              0.16485           2.6938         0.500     0.02002
0.99459            0.18619           0.36836
123456789012345                   67890     67890     67890     67890
CLARK-DICKINSON He-H2 AT 4 HBAR-OMEGA
```

19.8 TEST RUN heh2cdr2

He + H-H (harmonic oscillator with vibrational quantum number v)

-> He + H-H (harmonic oscillator with vibrational quantum number v')

This test run illustrates various other options available to the program. The input value of NPROP is 1, which means that the transpose of the overlap matrix is used rather than its inverse. This ensures a unitary scattering matrix. The use of LGS(15) to calculate eigenphase sums is used, which results in output to FORTRAN unit 11 and FORTRAN unit 16.

In order to run this test program, perform the following steps:

1) Ensure that the following files are available:

test/heh2cdr2.4	(input data for FORTRAN unit 4)
test/heh2cdr2.5	(input data for FORTRAN unit 5)
test/heh2cdr2.param	(include file of parameters copied to parameter.inc)
runtest.csh	(C Shell script to run program)
script/heh2cdr2.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)

2) Enter the command

```
runtest.csh heh2cdr2
```

3) Output files produced:

heh2cdr2.out	standard output (UNIT 6)
heh2cdr2.3	date and time of the run (UNIT 3)
heh2cdr1.7	Symmetrized Reactance Matrix (UNIT 7)
heh2cdr1.11	eigenphases at r (UNIT 11)
heh2cdr2.15	short output (UNIT 15)
heh2cdr2.16	eigenvectors (UNIT 16)

Input to FORTRAN unit 4 (for potential routine):

Identical to that of heh2r1

Input to FORTRAN unit 5 (for RMPROP 2.1.2):

```
HELIUM-HARMONIC OSCILLATOR COLLISION NPROP = 1
TEST SUITE RUN 7
FT
   15  800    1    1    2   10    1
    0  100    0    0    0
0.245080E+4       14    0
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0
    1   -1    0
   14
0.16016000      1.20              0
 0.0010        0.0010          6.50
    1
         0.1500
         0.1500
         0.1500
         0.1E-6
         0.1E-6
         0.1E-6
         3.0000
         3.0000
         3.0000
         0.1500
         1.0E+1
    0
 0.0
         1.E-30        5.0        1
FFFFFFFFF
TTFFFFFFT
         1.E-02     200.0
TFT    0    2     2
    0
   6.2
TTFFFTFFFFFFTFT
TTFFFTFFFFFFTFT
   3   3
 1.E-3           1.E-3
   3
   1   1   1
   2   3   4
```

19.9 TEST RUN hfhfpbsr1

```
 H-F (Morse oscillator with vibrational quantum number v1)
+
 H-F (spherically averaged Morse oscillator with vibrational quantum number v2)

-> H-F (Morse oscillator with vibrational quantum number v1')
+
 H-F (spherically averaged Morse oscillator with vibrational quantum number v2')
```

This test run is presented to show that the program can also do simultaneous second-case runs in cases where the potential is spherically symmetric, so that all of the L(I) read in are zero, and the individual orbital angular momenta all have the identical numerical value and are read in from FORTRAN unit 5 as elements of the input array OAM(I).

In order to run this test program, perform the following steps:

1) Ensure that the following files are all available:

test/hfhfpbsr1.4	(input data for FORTRAN unit 4)
test/hfhfpbsr1.5	(input data for FORTRAN unit 5)
test/hfhfpbsr1.param	(include file of parameters copied to parameter.inc)
runtest.csh	(C Shell script to run program)
script/hfhfpbsr1.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)

2) Enter the command

```
runtest.csh hfhfpbsr1
```

3) Output files generated

hfhfpbsr1.out	standard output (UNIT 6)
hfhfpbsr1.15	short output (UNIT 15)

Input to FORTRAN unit 4 (for potential routine):

```
TF
     1.1966302       0.216606    1731.74265        1.0
0    0    0
1    1    2    1    3    2    1    4    2    3
1    5    2    4    3
1    2    1    3    1    2    4    1    3    2
5    1    4    2    3
3
9    11    5
   4.0          20.0
```

(this is only the first nine (10) lines of FORTRAN unit 4: there are over two thousand (2000) lines of oscillator matrix elements not included here!)

Input to FORTRAN unit 5 (for RMPROP 2.1.2):

```
SPHERICALLY AVERAGED HF HF LTYPE2 SECOND ENERGY FOR OAM
TEST SUITE RUN 8
FT
   15 4000    3    1    2    1    2
    0    0    0   30    0
1.823454E+4       15    0
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0
    1   -2    0
   15
     0.05602128            3.0   72
     0.05602128            3.0   84
 0.0600        0.0600         100.0
    1
         1.0E-1
         1.0E-1
         1.0E-1
         0.06
         0.1E-5
         0.1E-5
         0.06
         3.4E+3
         3.4E+3
         1.0E+1
         3.0E+2
    0
    0.0
         1.E-03            75.0   0
TFFFFFFFFF
TTFFFFFFFT
         1.E-03      1000.0
TTF    0    2    0
0
80.0
TTFFFTFTFFFFFFF
FFFFFTFTFFFFFFF
   3   3   3
 1.E-3           1.E-3
   3
   1   1   1
   2   3   4
```

19.10 TEST RUN hfhfadr1

H-F (rigid rotor with rotational quantum number j1)
+ H-F (rigid rotor with rotational quantum number j2) ->

H-F (rigid rotor with rotational quantum number j1')
+ H-F (rigid rotor with rotational quantum number j2')

The purpose of this test run is to write out information for the next test run, which will act as a "later-date" second-case run. The next run will use exactly the same energy as this run, in order to demonstrate that later date second-case runs do in fact run correctly.

In order to run this test program, perform the following steps:

1) Ensure that the following files are available:

test/hfhfadr1.4	(input data for FORTRAN unit 4)
test/hfhfadr1.5	(input data for FORTRAN unit 5)
test/hfhfadr1.param	(include file of parameters copied to parameter.inc)
runtest.csh	(C Shell script to run program)
script/hfhfadr1.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)

2) Enter the command

```
runtest.csh hfhfadr1
```

3) Output files produced are

hfhfadr1.out	standard output (UNIT 6)
hfhfadr3.out	date and time of the run (UNIT 3)
hfhfadr1.8	restart information for hfhfadr2 (UNIT 8, nonASCII)
hfhfadr1.14	restart information for hfhfadr2 (UNIT 14, nonASCII)
hfhfadr1.15	short output (UNIT 15)

```
Input to FORTRAN unit 4 (for potential routine):

   1.733
    1
    1
    5
T
3
 TESTING LATER DATE SECOND ENERGY RUNS: REFERENCE ENERGY RUN
 JSUM=4
 ALEXANDER-DEPRISTO DIPOLE-DIPOLE RIGID ROTOR HF-HF PES
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0
    0    1    1    1    2    2    2    3    2    2
    2    3    3    4
    0    0    1    1    0    1    1    0    2    2
    2    1    1    0
    0    1    0    2    2    1    3    3    0    2
    4    2    4    4
    0    1    0    0
    9
    1    2    2    0    1    1    0    1    1
    0    1    1    0    1    1    1    2    2
    0    0    1    0    0    1    0    0    1
1744.604       0.01889699     1.733          0.0            T 1000.0
    1    1    1    0    3    1
    1    1    1    0    0
    1
    0
```

Input to FORTRAN unit 5 (for RMPROP 2.1.2):

```
HF-HF ALEXANDER-DEPRISTO INITIAL LATER DATE RUN
TEST SUITE RUN 9
FT
   14 4000    3    1    2    1    1
    0    5    0   30    0
1.823454E+4       14    0
    0    1    0    2    2    1    3    3    0    2
    4    2    4    4
    1   -1    1
   14
   0.0027929250       2.0         0
 0.0600        0.0600          15.0
    1
         1.0E-1
         1.0E-1
         1.0E-1
         0.06
         0.1E-5
         0.1E-5
         0.06
         3.4E+3
         3.4E+3
         10.0
         300.0
    0
 0.0
         1.E-03       50.0        2
TFFFFFFFFF
TTFFFFFFFT
         1.E-03    1000.0
FTF    0    2    0
    0
       9.0       13.0
TTTFFTFTFFFFFFFF
FTTFFTFTFFFFFFFF
   3   3
 1.E-3           1.E-3
   4
   1   1   1   1
   1   2   3   4
```

19.11 TEST RUN hfhfadr2

```
   H-F (rigid rotor with rotational quantum number j1)
 + H-F (rigid rotor with rotational quantum number j2)  ->

   H-F (rigid rotor with rotational quantum number j1')
 + H-F (rigid rotor with rotational quantum number j2')
```

The purpose of this test run is to show that the later-date second-case run option works correctly, reading in data from FORTRAN unit 8 and FORTRAN unit 14 created by the previous run. The potential, basis set, stepsize, and energy are all identical to those of the previous run.

In order to run this test program, perform the following steps:

1) Ensure that the following files are available:

test/hfhfadr2.4	(input data for FORTRAN unit 4)
test/hfhfadr2.5	(input data for FORTRAN unit 5)
test/hfhfadr2.param	(include file of parameters copied to parameter.inc)
runtest.csh	(C Shell script to run program)
script/hfhfadr2.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)
hfhfadr1.8	(intermediate data written by hfhfadr2 to FORTRAN unit 8)
hfhfadr1.14	(first-energy information written by hfhfadr1 to FORTRAN unit 14)
runtest.csh	(C Shell script to run program)

2) Enter the command

```
runtest.csh hfhfadr2
```

3) Output files produced are

hfhfadr2.out	standard output (UNIT 6)
hfhfadr2.3	the date and time of the run (UNIT 3)
hfhfadr2.8	restart information (UNIT 8, nonASCII)
hfhfadr2.14	restart information (UNIT 14, nonASCII)
hfhfadr2.15	short output (UNIT 15)

Input to FORTRAN unit 4 (for potential routine):

Identical to that from hfhfadr1.4

Input to FORTRAN unit 5 (for RMPROP 2.1.2):

```
HF-HF ALEXANDER-DEPRISTO SECOND LATER DATE RUN
TEST SUITE RUN 10
FT
   14 4000    3    1    2    1    2
    0    5    0   30    0
1.823454E+4       14    0
    0    1    0    2    2    1    3    3    0    2
    4    2    4    4
    1    2   -1
   14
   0.0027929250       2.0         0
   0.0027929250       2.0         0
 0.0600        0.0600          15.0
    1
         1.0E-1
         1.0E-1
         1.0E-1
         0.06
         0.1E-5
         0.1E-5
         0.06
         3.4E+3
         3.4E+3
         10.0
         300.0
    0
 0.0
         1.E-03       50.0        2
TFFFFFFFTF
TTFFFFFFFT
         1.E-03    1000.0
FTF    0    2    0
   0
      70.0
TTTFFTFTFFFFFFFF
FTTFFTFTFFFFFFFF
   3   3   3
 1.E-3           1.E-3
   4
   1   1   1   1
   1   2   3   4
```

19.12 TEST RUN atdir1

Atom + rigid diatomic molecule (j=0)-> Atom + rigid diatomic molecule (j' < 4)

This sample run solves the close coupling equations in three dimensions for a general atom-rigid rotor diatomic system with the interaction potential and basis set of Lester and Bernstein. This is one of the first systems for which published results were obtained for solution of the molecular close coupling equations, and it provides a standard historical benchmark in the field. The units originally used were "reduced" units, where energy, mass and distance were scaled with respect to the interaction potential and the mass of the diatomic molecule. The present subroutines have converted these to atomic units.

This run is a sample of the use of prevgr1.f and vgr1.f.

The run is performed at a single energy, with a single basis, and it saves restart information alternating between FORTRAN units 17 and 18. The asymptotic analysis is performed at 4 distances, and comparison is made of the magnitudes of all transition probabilities between the first four channels. The asymptotic boundary conditions are enforced by matching to sine/cosine boundary conditions using the asymptotic wave vectors, and the variable LTASY is .FALSE. (the transformation to the asymptotic basis is NOT performed).

In order to run this test program, perform the following steps:

1) Ensure that the following files are available:

test/atdir1.4	(input data for FORTRAN unit 4)
test/atdir1.5	(input data for FORTRAN unit 5)
test/atdir1.param	(include file of parameters copied to parameter.inc)
runtest.csh	(C Shell script to run program)
script/atdir1.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)

2) Enter the command

```
runtest.csh atdir1
```

3) Output files generated:

atdir1.out	standard output (UNIT 6)
atdir1.3	the date and time of the run (UNIT 3)
atdir1.15	short output (UNIT 15)

Input to FORTRAN unit 4 (for potential routine):

```
T                                                       LSTAP
    0    2    2    2    4    4    4    4    4    6     JIN
    6    6    6    6    6    6                         JIN
    2    2                                             IHOM,LAMMAX
```

Input to FORTRAN unit 5 (for RMPROP 2.1.2):

```
 ATOM-DIATOM LESTER AND BERNSTEIN
 TEST SUITE RUN 11
FF
   16 4000    1    1    1    1    2
    0  100    0    0    0
5.000000E+2       16    6
    6    4    6    8    2    4    6    8   10    0
    2    4    6    8   10   12
    1   -1    0
   16
   1.10000          0.67          0
0.0008         0.0008            9.00
    1
         0.1300
         0.1300
         0.1300
         8.0E-6
         8.0E-6
         8.0E-6
         8.0E-0
         8.0E-0
         8.0E-0
         1.4E+0
         0.5E+1
    0
 0.0
         1.E-03         100.0     2
TFFFFFFFTF
TTFFFFFFFT
         1.E-03        1000.0
FFF    0    2    0
   4
       5.0            6.0            7.0            8.0
TTTFFTFFTFFFFFFTFFF
TTTFFTFFTFFFFFFTFFF
   3   2   3
 1.E-3
  10
   1   1   1   1   2   2   2   3   3   4
   1   2   3   4   2   3   4   3   4   4
```

19.13 TEST RUN ehydr1

Electron + hydrogen atom scattering

The purpose of this test run is to illustrate the capability of the program to perform atom-electron scattering. In particular this run propagates the full R matrix (inhomogeneous option). It is important to note that the first sector begins at r = 1.0 x 10**-6. The run uses the inverse of the overlap matrix rather than its transpose, so that the unitarity of the scattering matrix provides a sensitive check of whether the overlap matrix is orthogonal. The run also performs calculations at a single energy with a single basis set, and it prints out the full R matrix at each step (this generates a good deal of output). Note also that the energy of the run is included in FORTRAN unit 4 as well as in FORTRAN unit 5 (we have an energy-dependent potential).

In order to run this test program, perform the following steps:

1) Ensure that the following files are all available:

```
test/ehydr1.4         (input data for FORTRAN unit 4)
test/ehydr1.5         (input data for FORTRAN unit 5)
test/ehydr1.param     (include file of parameters copied to parameter.inc)
runtest.csh           (C Shell script to run program)
script/ehydr1.csh     (C Shell script to manage files for this test run)
script/Makefile.mak   (file for UNIX make utility to perform compilation)
```

2) Enter the command

```
runtest.csh ehydr1
```

3) Output files generated

```
ehydr1.out            standard output (UNIT 6)
ehydr1.15             short output (UNIT 15)
```

Input to FORTRAN unit 4 (for potential routine):

```
   1
FFTFF
T
2.0
1
THIS IS A TEST RUN FOR ELECTRON-HYDROGEN ATOM SCATTERING
```

Input to FORTRAN unit 5 (for RMPROP 2.1.2):

```
ELECTRON-HYDROGEN ATOM TEST RUN 2 CHANNELS
TEST RUN FOR ALL 4 ELEMENTS OF R MATRIX R1MT R2MT R3MT R4MT
FF
    2 4000   13    1    2   10    1
    0  300    0   30    0
1.0                2    1
    0    0
    1   -1    0
    2
2.000000000            1.0E-6     1
        1.0E-8         1.0E-8     19.5
    1
         5.0D-2
         0.0001
         0.0001
         0.1D-9
         0.1D-9
         0.1D-9
         2.0D+4
         2.0D+4
         2.0D+4
         1.0D+2
         9.0D+3
    0
 0.0
         1.D-03        39.0       0
FFFFFTFFTT
TTFFFFFFFT
         1.D-03      200.0
TFT    0    0    0
    0
380.0          390.0
TTFFTTTTFFFFFFFF
TTFFTTTTFFFFFFFF
   3   3
 0.D-3           0.D-3
   3
   1   1   1
   2   3   4
```

19.14 TEST RUN hei2r1

He + I_2 (vibrating rotor with v=0, various j)
-> He + I_2 (vibrating rotor with v=0, various j')

The purpose of this run is to illustrate the expansion of an arbitrary vibrational potential for an atom-diatom potential in a set of Legendre functions with appropriate (R dependent) expansion coefficients. In other words this run differs from test run 11, atdir1, because of the dependence of the potential on the bond length of the I_2 molecule. This is a sample of the use of prevgr2.f and vgr2.f.

This run is performed at a single energy and with a single basis, using the homogeneous option.

In order to run this test program, perform the following steps:

1) Ensure that the following files are all available:

test/hei2r1.4	(input data for FORTRAN unit 4)
test/hei2r1.5	(input data for FORTRAN unit 5)
test/hei2r1.param	(include file of parameters copied to parameter.inc)
test/hei2r1.expand	(include file of parameters copied to expand.inc)
runtest.csh	(C Shell script to run program)
script/hei2r1.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)

2) Enter the command

```
runtest.csh hei2r1
```

3) Output files generated

hei2r1.out	standard output (UNIT 6)
hei2r1.15	short output (UNIT 15)
hei2r1.3	date and time of run (UNIT 3)

```
Input to FORTRAN unit 4 (for potential routine):

5.0387
    3
    3    3    3
   23   13   11
 4.99            9.99
T
    2
He + I2
USE CHEM PHYS LET POTENTIAL FUNCTION
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    1    1    1
    1    1    1    1    1    1    1    1    1    1
    1    1    2
    0    2    2    4    4    6    6    8    8   10
   10   12   12   14   14   16   16    0    2    2
    4    4    6    6    8    8   10   10   12   12
   14   14    0
    0    1    0
115662.5447    0.0009773422867    5.0387          0.0       F 1000.0            1
   70   40   10    7    3    1
  400    4    4    0    2
    3
    5
    6
    0
   12
    7
    6
```

Input to FORTRAN unit 5 (for RMPROP 2.1.2):

```
SCHWENKE-TRUHLAR SURFACE FOR He-I2 CPL LETT
TEST OF RMPROP 2.1.2 FOR ATOM-VIBRATING ROTOR
FT
   33 4000   13    1    2   10    1
    0  300    0   30    0
7.183007D+3       33    1
    1    1    3    3    5    5    7    7    9    9
   11   11   13   13   15   15   17    1    1    3
    3    5    5    7    7    9    9   11   11   13
   13   15    1
    1   -1    0
   33
0.00150527503          3.00       0
        0.0400        0.0400        200.0
    1
         0.7E-1
         0.7E-1
         0.7E-1
         4.0E-2
         0.1E-5
         0.1E-5
         4.0E-2
         1.0E+1
         3.4E+0
         1.0E+1
         5.0E+2
    0
0.0
         1.D-32        40.0       1
FFFFFFFFF
TTFFFFFFT
         1.D-02     1000.0
TTF    0    2    0
    0
180.0          190.0
TTFFTTFTFFFFFFFF
TTFFTTFTFFFFFFFF
   3   3
 0.D-3           0.D-3
   3
   1   1   1
   2   3   4
```

19.15 TEST run heh2tkr1.

The He-H2 vibrating rotor calculation using the Tsapline-Kutzelnigg-Raczkowski potential and the basis set from Raczkowski, Lester, and Miller.

This test run is a vibrating rotor utilizing the prevgr2.f and vgr2.f set of modular potential routines. The asymptotic eigenvalues are generated by SUBROUTINE WEIGH. No homonuclear symmetry is used for this run, to compare with the heh2tkr2 test run which is identical except that homonuclear symmetry is utilized.

This calculation has been repeated with entirely different codes and potential matrix routines (de Vogelaere's method and finite differencing are the two other close-coupling methods), and the final, well-checked transition probability matrix is given in Section 19.16, after the input for heh2tkr2.

In order to run this test program, perform the following steps:

1) Ensure that the following files are all available:

test/heh2tkr1.4	(input data for FORTRAN unit 4)
test/heh2tkr1.5	(input data for FORTRAN unit 5)
test/heh2tkr1.param	(include file of parameters copied to parameter.inc)
test/heh2tkr1.expand	(include file of parameters copied to expand.inc)
runtest.csh	(C Shell script to run program)
script/heh2tkr1.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)

2) Enter the command

```
runtest.csh heh2tkr1
```

3) Output files generated

heh2tkr1.out	standard output (UNIT 6)
heh2tkr1.15	short output (UNIT 15)
heh2tkr1.3	date and time of run (UNIT 3)

Input to fortran unit 4:

```
1.4016103               RE
    1                   numq
    5    9    9         nquad
   12   13   11         nquada
T                       lstap
    2                   nlin
He + H2
USE TSAPLINE-KUTZELNIGG-RACZKOWSKI POTENTIAL FUNCTION
    0    0    0    0    0    1    1    1    1    2      v
    2    2
    0    2    4    6    8    0    2    4    6    0      j
    2    4
    1    0    1             ihom,ivibj,ieign
 918.07575     0.01989366     1.4016103      0.0            F  100.0
   70   40    5    0    3    1       nquad,nbasis,npts,iprint,inode,iwght
  400    4    4    8    0            nint,nl,nsave,jsave,ihom
```

```
    5                                 npts
    0                                 npts
    5                                 numt

Input to fortran unit 5:

raczkowski-lester SURFACE FOR He-H2 CPL LETT
TEST OF RMPROP 2.1.2 FOR ATOM-VIBRATING ROTOR
FT
   12 5000    1    1    1   10    1
    0  300    0   30    0
0.245080D+4       12    0
    0    2    4    6    8    0    2    4    6    0
    2    4
    1   -1    0
   12
0.03307413000          1.00       0
        0.0010        0.0010        018.0
    1
         5.0E-2  epsa
         0.7E-1  epsb
         0.7E-1  epsc
         1.0E-3  hmina
         0.1E-5  hminb
         0.1E-5  hminc
         4.0E+0  hmaxa
         1.0E+1  hmaxb
         3.4E+0  hmaxc
         2.5E+1  repsa
         5.0E+2  repsb
    0
0.0
         1.D-32        40.0       1
FFFFFFFFF
TTFFFFFFT
         1.D-02       50.0
TTF    0    2    2
    3
  8.0           10.0           12.0
TTFFTTFTFFFFFFF
TTFFTTFTFFFFFFF
   3   3
 1.D-3           0.D-3
   7
   1   1   1   1   1   1   1
   1   2   3   4   5   6   7
```

19.16 TEST run heh2tkr2.

This test run is identical to heh2tkr1, except that homonuclear symmetry is exploited. The only difference is in the input to fortran unit 4, given below. Transition probabilities should be the same as for heh2tkr1.

In order to run this test program, perform the following steps:

1) Ensure that the following files are all available:

test/heh2tkr2.4	(input data for FORTRAN unit 4)
test/heh2tkr2.5	(input data for FORTRAN unit 5)
test/heh2tkr2.param	(include file of parameters copied to parameter.inc)
test/heh2tkr2.expand	(include file of parameters copied to expand.inc)
runtest.csh	(C Shell script to run program)
script/heh2tkr2.csh	(C Shell script to manage files for this test run)
script/Makefile.mak	(file for UNIX make utility to perform compilation)

2) Enter the command

```
runtest.csh heh2tkr2
```

3) Output files generated

heh2tkr2.out	standard output (UNIT 6)
heh2tkr2.15	short output (UNIT 15)
heh2tkr2.3	date and time of run (UNIT 3)

Input to Fortran Unit 4:

```
1.4016103                RE
    1                    numq
    5    9    9          nquad
   12   13   11          nquada
T                        lstap
    2                    nlin
He + H2
USE TSAPLINE-KUTZELNIGG-RACZKOWSKI POTENTIAL FUNCTION
    0    0    0    0    0    1    1    1    1    2      v
    2    2
    0    2    4    6    8    0    2    4    6    0      j
    2    4
    0    0    1             ihom,ivibj,ieign
 918.07575     0.01989366     1.4016103      0.0            F  100.0
   70   40    5    0    3    1       nquad,nbasis,npts,iprint,inode,iwght
  400    4    4    8    2            nint,nl,nsave,jsave,ihom
    5                               npts
    0                               npts
    3                               numt
```

Input to Fortran Unit 5:
Identical to that for heh2tkr1.

Transition probabilities P(M,N) at total energy E = 0.03307413000 hartrees

M-N	1	2	3	4	5	6	7
1	4.380764E-01	5.132297E-01	4.836745E-02	3.265070E-04	1.681079E-08	1.194160E-08	3.631780E-10
2	5.132297E-01	2.686184E-01	2.146308E-01	3.520715E-03	3.042534E-07	4.995453E-08	7.498562E-09
3	4.836745E-02	2.146308E-01	6.910074E-01	4.598611E-02	8.070278E-06	1.159049E-07	3.608936E-08
4	3.265070E-04	3.520715E-03	4.598611E-02	9.495436E-01	6.229709E-04	1.359766E-08	6.375251E-08
5	1.681079E-08	3.042534E-07	8.070278E-06	6.229709E-04	9.993686E-01	3.048231E-11	3.734263E-10
6	1.194160E-08	4.995453E-08	1.159049E-07	1.359766E-08	3.048231E-11	9.179298E-01	8.207002E-02
7	3.631780E-10	7.498562E-09	3.608936E-08	6.375251E-08	3.734263E-10	8.207002E-02	9.179299E-01

20. DESCRIPTION OF FILES IN VERSION 2.1.2

Version 2.1.2 of the RMPROP code consists of the files listed below. The files are organized by distribution package directories and also by certain subclasses of filetype. In addition, we have given notations to find the appropriate section of the manual for further documentation about files where such notation would be appropriate. The file rmprop2.doc is this on-line manual for the current version 2.1.2. The RMPROP program source code and the necessary subroutines are contained in the files rmprop2.f, exscat2.f, and gnscat2.f.

RMPROP2.1.2/	Main directory of distribution package
rmprop2.doc	This online manual
runtest.csh	Main C SHELL script to run test suite elements
script	Directory with scripts and Makefile.mak
src	Directory with source code
test	Directory with test suite input, sample output, and include files

Contents of DIRECTORY script

Makefile.mak	Makefile for RMPROP 2.1.2 (see Section 19)
atdir1.csh	Scripts for the test suite elements
ehydr1.csh	
heh2cdr1.csh	
heh2cdr2.csh	
heh2tkr1.csh	
heh2tkr2.csh	
hehfr1.csh	
hei2r1.csh	see Sections 19-19.16
hfhfadr1.csh	
hfhfadr2.csh	
hfhfpbsr1.csh	
hfhfstr1.csh	
hfhfstr2.csh	
hfhfstr3.csh	
hfhfstr4.csh	
hfhfstr5.csh	

Contents of DIRECTORY src

Main program code: (See Section 16)

rmprop2.f	Main driver
exscat2.f	Extra asymptotic routines
gnscat2.f	Asymptotic analysis; other miscellaneous routines

Machine-specific codes: (See Sections 6.1, 16.2-16.2.4)

sdralpha.f	DEC ALPHA AXP 3000/M500
sdrcray.f	Cray UNICOS
sdrrs6000.f	IBM RS/6000
sdrsun.f	Sun Sparcstation
dateclock.c	C Shell script to get date and time on IBM RS/6000

Version-specific codes: (See Sections 5.3, 6.2-6.8, and 17)

verce.f	For Version 2.1.1ce (Cray-2 MXMA, MINV, EVCRSP)
vercs.f	For Version 2.1.1cs (Cray-2 MXMA, MINV, RS)
veri.f	For Version 2.1.1i (IBM RS/6000 DGEMM, DGETRF/S, EVIRSP)
verp.f	For Version 2.1.1 (Portable, DGEMM, DGETRF/S, EVDRSP)
verxe.f	For Version 2.1.1xe (Cray C90 SGEMM, SGETRF/S, EVCRSP)
verxs.f	For Version 2.1.1xs (Cray C90 SGEMM, SGETRF/S, RS)

Potential energy function routines and associated subroutines:
(See Sections 10, 11, 19)

cleb2.f	Calculates Clebsch-Gordan coefficients by recursion
fcoef.f	Calculates angular integrals for st potential and vgr3.f
fcoef5.f	Calculates angular integrals for st potential and vgr3.f
+ JTOT=0 only	
flambda.f	Calculates Percival-Seaton coefficients
gaussq.f	General routine to perform Gaussian quadratures over angles and vibrational coordinates
hfmsptntl.f	"User-supplied" FUNCTION PTNTL to be used with test runs in 19.1-19.5
hfwreigen.f	"User-supplied" FUNCTION EIGEN to be used with test runs 19.1-19.5
i2gnptntl.f	"User-supplied" FUNCTION PTNTL to be used with test runs 19.14
jsymbol.f	Calculates 3-J, 6-J, and 9-J symbols
leg.f	Generates Legendre polynomials
potatdi.f	"User-supplied" subroutines test run 19.12
potehyd.f	Subroutines PREPOT and POT for test run 19.13
potheh2.f	Subroutine POT for test runs 19.7 - 19.8
pothehf.f	"User-supplied" Subroutine POT (test run 19.6)
pothei2.f	"User-supplied" Subroutines for (test run 19.14)
pothfhfad.f	"User-supplied" Subroutines for (test runs 19.10 - 19.11)
pothfhfpbs.f	Subroutine POT for test run 19.8
preheh2.f	Subroutine PREPOT for test runs 19.7 - 19.8
prehfhfpbs.f	Subroutine PREPOT for test run 19.9
prevgr2.f	Subroutine PREPOT for general 3-D atom-diatom (test run 19.14 - 19.16) potential matrix routine
prevgr3.f	Subroutine PREPOT for general 3-D diatom-diatom potential matrix routine (test runs 19.10 - 19.11)
rigptntl.f	"User-supplied" FUNCTION PTNTL; dummy routine to be used with weigh.f for vgr2.f and vgr3.f when the diatom is a rigid rotor
stpot.f	Subroutine PREPOT and POT for test runs 19.1-19.5
stpot1.f	Expansion coefficients for angular portion of interaction potential matrix for test runs 19.1-19.5
stpot2.f	Vibrational portion of for test runs 19.1-19.5

vgr1.f	General 3-D atom-rigid diatom potential matrix routine
vgr2.f	General 3-D atom-diatom potential matrix routine
vgr3.f	General 3-D diatom-diatom potential matrix routine
wdata.f	Reads weights and nodes for vibrational quadratures created by weigh.f
weigh.f	Calculates weights and nodes for vibrational quadratures
xmat.f	Harmonic oscillator matrix elements of x**p for heh2
yqqm.f	Calculates normalized Launay functions for vgr3.f

Contents of DIRECTORY test

Input files: See Sections 11.2.2, 11.3.3, 11.3.4, 11.4.3 and 11.4.4 for descriptions of *.4 files and Section 13 for descriptions of *.5 files. Section 19 has descriptions of each input file for each test run.

atdir1.4
atdir1.5
ehydr1.4
ehydr1.5
heh2cdr1.4
heh2cdr1.5
heh2cdr2.4
heh2cdr2.5
heh2tkr1.4
heh2tkr1.5
heh2tkr2.4
heh2tkr2.5
hehfr1.4
hehfr1.5
hei2r1.4
hei2r1.5
hfhfadr1.4
hfhfadr1.5
hfhfadr2.4
hfhfadr2.5
hfhfpbsr1.4
hfhfpbsr1.5
hfhfstr1.4
hfhfstr1.5
hfhfstr2.4
hfhfstr2.5
hfhfstr3.4
hfhfstr3.5
hfhfstr4.4
hfhfstr4.5
hfhfstr5.4
hfhfstr5.5

Include files: See Sections 10.2, 11.3.2, 11.4.2, 14, and 15 for discussions
on *.param include files and Sections 11.3.2 and 11.4.2 for
discussions of *.expand files.

heh2tkr.expand
hei2r1.expand
hfhfadr1.expand
hfhfadr2.expand
atdir1.param
ehydr1.param
heh2cdr1.param
heh2cdr2.param
heh2tkr1.param
heh2tkr2.param
hehfr1.param
hei2r1.param
hfhfadr1.param
hfhfadr2.param
hfhfpbsr1.param
hfhfstr1.param
hfhfstr2.param
hfhfstr3.param
hfhfstr4.param
hfhfstr5.param

Sample Short Output files: from test suite for debugging purposes.
These are from the distribution version run on the Cray C-90.
hfhfstr1.15
hfhfstr2.15
hfhfstr3.15
hfhfstr4.15
hfhfstr5.15
hehfr1.15
heh2cdr1.15
heh2cdr2.15
hfhfpbsr1.15
hfhfadr1.15
hfhfadr2.15
atdir1.15
ehydr1.15
hei2r1.15
heh2tkr1.15
heh2tkr2.15

21. BIBLIOGRAPHY

1. J. C. Light and R. B. Walker, J. Chem. Phys. 65, 4272 (1976).

2. N. M. Harvey, (nee N. A. Mullaney) Ph.D. Dissertation, University of Minnesota, Minneapolis, 1979.

3. D. G. Truhlar, N. M. Harvey, K. Onda, and M. A. Brandt, in "Algorithms and Computer Codes for Atomic and Molecular Scattering Theory", Vol. 1, edited by L. Thomas, National Resource for Computation in Chemistry, Lawrence Berkeley Laboratory, Berkeley, CA, 1979, pp. 220-289.

4. D. W. Schwenke and D. G. Truhlar, in "Supercomputer Applications", edited by R. W. Numrich, Plenum, New York, 1985, pp. 215-254.

5. D. W. Schwenke and D. G. Truhlar, J. Chem. Phys. 88, 4800 (1988).

6. L. A. Collins and N. F. Lane, Phys. Rev. A 14, 1358 (1976).

7. N. A. Mullaney and D. G. Truhlar, Chem. Phys. 39, 91 (1979).

8. N. M. Harvey and D. G. Truhlar, Chem. Phys. Lett. 74, 252 (1980).

9. L. L. Poulsen, G. D. Billing, and J. I. Steinfeld, J. Chem. Phys. 68, 5121 (1978).

10. M. H. Alexander and A. E. DePristo, J. Chem. Phys. 65, 5009 (1976).

11. W. A. Lester Jr. and R. B. Bernstein, Chem. Phys. Lett. 1, 207 (1967). Note however the errata on pp. 347-348.

12. D. G. Truhlar and N. A. Mullaney, J. Chem. Phys. 68, 1574 (1978).

13. D. W. Schwenke and D. G. Truhlar, Chem. Phys. Lett. 98, 217 (1983).

14. D. W. Schwenke and D. G. Truhlar, J. Chem. Phys. 83, 3454 (1985).

15. B. Tsapline and W. Kutzelnigg, Chem. Phys. Lett. 23, 173 (1973).

16. A. W. Raczkowski and W. A. Lester, Chem. Phys. Lett. 47, 45 (1977), and (E) Chem. Phys. Lett. 49, 398 (1977).

22. ERRATA FOR MANUAL OF RMPROP VERSION 1.0

The following is a list of errata for the on-line manual for RMPROP 1.0 (a program for quantum inelastic scattering, found in MOTECC-91, edited by Enrico Clementi).

1) The discussion of LPSYM(1). The manual for RMPROP 1.0 says, "...if .TRUE., test the symmetry of the R matrix at each sector for each energy just before entering subroutine RPROP."

This contains a typographical error.

The manual for RMPROP 1.0 should say,
"...if .TRUE., test the symmetry of the R^4 matrix at each sector for each energy just before entering subroutine RPROP."

2) The discussion of LPSYM(2). The manual for RMPROP 1.0 says, "...if .TRUE., test the orthogonality of the TAUE matrix {7.5} at each sector for each energy just after initial call to subroutine TAUMTS."

This is inaccurate.

The manual for RMPROP 1.0 should say,
"...if .TRUE., test the orthogonality of the TAUE matrix {7.5} at each energy at each sector for all r > RBIG just after initial call to subroutine TAUMTS. This option will only work for LTYPE2 second-case runs."

3) The discussion of LPSYM(4). The manual for RMPROP 1.0 says, "...if .TRUE., upon entrance to subroutine GNSCAT, calculate the maximum relative difference and the average relative difference of the off-diagonal elements, and print out the R matrix in the mixed basis {7.9}."

This is inaccurate.

The manual for RMPROP 1.0 should say,
"...if .TRUE., upon entrance to subroutine GNSCAT, calculate the maximum relative difference and the average relative difference of the off-diagonal elements of the raw R matrix, and print out the raw R^4 matrix."

4) The discussion of LPSYM(5). The manual for RMPROP 1.0 says, "...if .TRUE., print out the R^4 matrix transformed to the asymptotic basis {7.9}, and print out the maximum relative difference and average relative difference of the off-diagonal elements."

This contains a typographical error.

The manual for RMPROP 1.0 should say,
"...if .TRUE., print out the R^4 matrix transformed to the asymptotic basis {7.9}, and print out the maximum relative difference and average relative difference of the off-diagonal elements."

5) The instructions for performing elements of the test suite. All except the first element of the test suite say,

" runtest ..." where the ... is the name of the individual test.

This is incorrect.

The manual is correct for the FIRST element of the test suite, which says, "runtest.csh hfhfstr1".
The manual should be corrected to read similarly for the other elements of the test suite. That is, it should read,
"runtest.csh XXX" for ALL test runs where XXX is the name of the test run.

23. ERRATA FOR RMPROP: VERSION 1.0 CODE

The following is a list of known FORTRAN errors in RMPROP 1.0.

1) For LPR(4)= .TRUE., the legend written to FORTRAN unit 6 states that the accompanying matrix contains the "ELEMENTS OF PACKED INTERACTION MATRIX". This is inaccurate. It should print instead "ELEMENTS OF UNPACKED INTERACTION MATRIX". Note that the on-line manual does not specify whether the interaction matrix is in packed or unpacked form. This bug has been corrected in RMPROP-Version 2.0.

2) At the end of a run, the FORTRAN source code contains an instruction to close FORTRAN unit 6. This resulted in an error message on the IBM 3090. This statement has been deleted in RMPROP-Version 2.0.

3) In the subroutine stpot.f, the index for the DO-LOOP labeled 6 in subroutine PREPOT was mistyped as NVIB instead of NVIBX. This routine is used in the test runs hfhfstr1, hfhfstr2, hfhfstr3, and hfhfstr4. This error will affect the channel reordering according to energy in cases where the two channels whose indices are to be exchanged have different values for v_1 and v_2, but did not affect any of the results in the test suite. This bug has been corrected in RMPROP-Version 2.0.

4) In subroutine EXSCAT, the calculation of the eigenvectors associated with the individual eigenphases used to calculate the eigenphase sum of the S matrix, which was called by LPEPS/LGS(15), was performed incorrectly. This bug has been corrected in RMPROP-Version 2.0.

5) The legends generated for execution times of various tasks generated by LPR(1) in RMPROP-Version 1.0 wrote the names of the Cray SCILIB routines to FORTRAN unit 6 for all versions of the program. The names of the routines printed out have been changed to reflect the new matrix utilities used in RMPROP-version 2.0.

6) The call to SUBROUTINE HEADER has been removed from PROGRAM RMPROP, so that it is only called by SUBROUTINE PREPOT for each test run. As a result, the information printed out by SUBROUTINE HEADER will only appear once in FORTRAN unit 15 (the same information had been printed twice).

7) The input variable EPSDR was not passed to subroutine GNSCAT to allow the dropping of certain channels from the asymptotic analysis even if they had been propagated. This bug has been corrected in RMPROP-Version 2.0.

8) The call to SUBROUTINE POT from within SUBROUTINE MATCH was incorrect. This did not affect the results of any of the test runs in RMPROP-Version 1.0. This bug has been corrected in RMPROP-Version 2.0.

9) For several of the test runs in RMPROP: Version 1.0, the scattering matrix elements obtained on the IBM differed slightly from those obtained on the Cray-2. This is because of the structure of IF...GO TO blocks within the individual potential matrix routines. In particular, the order of the numerical quadratures for the interaction potential matrix elements (Section 11)

was selected by comparing the radial separation to an input value. On the IBM, the value stored by the computer for the radial separation differed by a very small amount (10**-20, "garbage") from that on the Cray, with the result that the switch between the different orders of numerical quadrature happened one sector before or later on the IBM than they did on the Cray-2. This bug has been corrected in RMPROP-Version 2.0, by re-writing the numerical test of the IF...GO TO blocks involved.