MESMER

Master **E**quation **S**olver for **M**ulti-**E**nergy well **R**eactions

Version 2.0

User's Manual

Struan H. Robertson, David R. Glowacki, Chi-Hsiu Liang,

Chris Morley, Robin Shannon, Mark Blitz and Michael J. Pilling

Last updated: 24 March 2012

1 Acknowledgements

This work was made possible through the help of several people not included as authors. The MESMER project is the culmination of several years of ME research conducted within the Leeds group, and we are pleased that we finally have something that is starting to resemble a general Master Equation program. We would like to acknowledge the following individuals: Dr. Nicholas Green, Dr. Kevin Hughes, and Dr. David Waller. Much of the MESMER development was carried out under the auspices of grants from the EPSRC and NERC. Finally, we would like to thank you, the users. No program is ever perfect, and while we have carried out tests with MESMER on several different chemical systems, we will not have caught every bug. Some of you will no doubt find yourselves particularly adept at breaking the program, and we hope you will provide us feedback. MESMER will only become a better program with your help.

2 Contents

3 What's New in MESMER 2.0

- The ability to constrain parameters together during a fitting exercise.

4 What Was New in MESMER 1.0

- Implementation of the Marquardt non-linear least squares algorithm for fitting experimental data.
- Analysable data extended to include experimental yields and eigenvalues.
- Variable parameter set extended to include transition state imaginary frequency and the exponential temperature dependence parameter of $\langle \Delta E \rangle_d$.
- An implementation of the WKB tunneling method.
- Quantum mechanical hindered rotor calculations extended to include asymmetric potentials.
- A plug-in class has been implemented that calculates thermodynamic tables for species defined in the molecular list.
- A plug-in class has been added to allow the input of pre-defined coupled states.

5 What Was New in MESMER 0.2

- Asymmetric rotor quantum mechanical energy levels.

- An abstract base class for energy transfer models has been added allowing different energy transfer specifications to be implemented.

- A plug-in class has been implemented for the exponential down model.

- A plug-in class for hindered rotation has been implemented. This class calculates the quantum mechanical energy levels of a hindered rotor in a basis of a one dimensional rotational eigenfunctions.

- An abstract base class for calculation methods has been added, allowing alternative calculation tasks to be implemented.

- A plug-in class has been added that to fit experimental data, by optimizing a merit function using a combination of golden section line searches and conjugate direction methods.

- A plug-in class for calculating microcanonical rate coefficients for non-adiabatic transitions.

6 Introduction

The modelling of unimolecular systems has application in a variety of environmental and industrial contexts. In recent years a great deal of progress has been made in the understanding and modelling of unimolecular systems over a range of temperatures and pressures The quantities of particular interest are the rate coefficients, time dependent species profiles, product yields, and branching ratios of the system being investigated. Each of these quantities typically shows a complex dependence on pressure and temperature. The modelling of industrial or environmental processes often involves conditions that are difficult to access experimentally, so it is important to be able to generate experimentally validated rate coefficient models that may be extrapolated to the sorts of conditions of interest in larger scale simulations. For example, typical experimental kinetics measurements of reactions important in the atmosphere are performed at very low pressures; however, in the atmosphere, these reactions occur at high pressures – so the low pressure results must be extrapolated to the higher pressures characteristic of the lower atmosphere. The use of stochastic techniques for describing the evolution of unimolecular systems – in particular the master equation (ME) – is a common means of linking laboratory studies and larger scale modelling. MESMER uses matrix techniques to formulate and solve the energy grained master equation (EGME) for unimolecular systems composed of an arbitrary number of wells, transition states, sinks, and reactants.

A unimolecular system is characterised by one or more potential wells (local minima) on the potential energy surface (henceforth, PES) which describes the energy of the atoms as a function of position. Each well represents a (meta-) stable species that can, in principle, be isolated. Wells are connected by transition states (TS) and a species in one well may be converted to another by passing through the TS that connects the wells. In many systems the TS can be associated with a saddle point on the system PES and so there is an energy barrier to inter-conversion of species. Thus, to convert from one species to another, the reactant must be activated – i.e., energy must be supplied to overcome the barrier separating the two wells. Typically, energy is supplied through collisions with bath gas molecules - the reactant will undergo a number of collisions with bath gas molecules some of which will be activating (net

increase in reactant energy) and some of which will be deactivating (net decrease in reactant energy). Since collision events and the amount of energy transferred are random quantities, the energy transfer process can be regarded as a random walk, and treated using techniques from stochastic process theory.

MESMER has been designed to improve upon and generalize previous developments in older master equation codes. MESMER has been written to offer a flexible approach to ME treatments of more complex systems. We have also attempted to incorporate various facilities that make it easy to apply the ME to gas kinetics. Some of the design goals which we had in mind while writing MESMER include:

1. Use standard, off-the-shelf technologies, so that the code may be readily maintained and extended. For example, we developed MESMER using the Microsoft's VS2008 integrated development environment, we use XML data representation for the input stream, we use Firefox as a PES viewer to aid in the construction of input files, and current developments are underway to increase compatibility between MESMER and such open source projects as OpenBabel.
2. Use open source C++ to write well-structured, object-oriented, cross-platform code that may be easily maintained and built upon by future developers. Where possible, we have designed the code so that future developers may be able to add increasing functionality via the use of plug-in classes. In addition, the code is commented, with references that indicate the methodologies used.

6.1 Citation

If you published results using MESMER, we would appreciate it if you would cite us. We are currently preparing a paper in which we detail some of the more interesting things we have implemented in MESMER, and we will ask you to cite that paper, but for the moment, please cite as follows:

Robertson, S. H.; Glowacki, D. R.; Liang, C.-H.; Morley, C.; Shannon, R.; Blitz, M.; Pilling, M. J., MESMER (Master Equation Solver for Multi-Energy Well Reactions), 2008-2012; an object oriented C++ program for carrying out ME calculations and eigenvalue-eigenvector analysis on arbitrary multiple well systems. http://sourceforge.net/projects/mesmer.

6.2 Accessing MESMER

MESMER is hosted by SourceForge website and can be accessed either by using the search facility provided or following the link:

http://sourceforge.net/projects/mesmer/

There are a number of tracker facilities that allow one to enter bugs and features requests and we strongly encourage this. If you wish to receive an email notification of items being added to these trackers, subscribe to the mesmer-notify mailing list which can also be found on the MESMER SourceForge project site.

7 Compilation and Execution

MESMER has been designed to be cross platform, and we have compiled it under Windows, LINUX/UNIX, and Mac operating systems, on both 32 and 64-bit architectures. The installation details describe how to get started using MESMER on several of these platforms. In general, we do our development and debugging on Windows. For large production runs, where we may want to run several hundreds or even thousands of ME calculations to explore the sensitivity of the results over the model parameter space, we tend to use LINUX.

7.1 Windows

7.1.1 Installing the Binary on Windows

Download the Windows installer and execute it. It will ask you to accept the LGPL licence, and to input the folder where you want it installed. It sets the PATH, MESMER_DIR and possibly MESMER_AUTHOR environment variables. An item is added to StartMenu/Programs with links to the documentation, but actually running MESMER would normally be done from a DOS command window. To remove MESMER from your computer, use Add and Remove Programs or click the StartMenu/Programs item Uninstall.

7.1.2 Compiling it yourself on Windows

If you *want* to build MESMER yourself, if an executable release is not available so that you *have* to build it yourself, or if you want to develop the code yourself, then we recommend the use of Visual C++. MESMER has been developed using Microsoft's Visual C++ 2008/2010 integrated development environment. Building and developing MESMER can be done using either the free Visual C++ 2010 Express Edition or the full version of Visual Studio (which isn't free). .

To build MESMER using VS2010, you could download the MESMER tar.gz distribution as described in section 7.2, but we would recommend using SVN (possibly with Tortoise SVN) when you will have the opportunity to use either the most recent development code by checking out

http://mesmer.svn.sourceforge.net/viewvc/mesmer/trunk/

or using a released version like

http://mesmer.svn.sourceforge.net/viewvc/mesmer/tags/Release_1.0.

To build it, go to the `\Windows VC10` folder, and use Visual C++ to open the file `MESMER.sln`. Clicking on the VS2010 command `Build the Solution` will build the binary executable in `\Windows VC10\Mesmer` folder. (Depending on the MESMER release that you download, you may see a `\Windows VC8` folder. This folder contains the VS2005 project and solution files for MESMER.)

7.1.3 Running on Windows

Open a DOS command window, most conveniently in the folder containing the data file. If the Windows installer has been used, or if the folder containing mesmer.exe has been manually added to the PATH environment variable, MESMER may be called as follows:

```
mesmer filename.xml
```

where `filename.xml` refers to the input XML file described below. See section 7.4 for a more complete description of the options and syntax of the command line.

7.2 Linux/UNIX/Mac

The first step to using MESMER is downloading it from the SourceForge project website. The downloaded release is distributed using `tar.gz` compression, which retains the directory structure. To uncompress the files in Windows, you can use free software like WinRAR. Under Linux/UNIX/Mac, you type the following command:

```
tar xvfz filename.tar.gz
```

where `filename` is the name of the particular MESMER release that you have downloaded.

Linux/UNIX compilation involves three easy steps: (1) compile the TinyXML libraries, (2) compile the QD libraries, and finally (3) compile the main executable. These steps are described sequentially below.

7.2.1 Compiling TinyXML

To compile TinyXML, which is what MESMER uses for input/output, the library has to be created by typing the following command under the `/tinyxml` folder:

```
make -f MakeLib DEBUG=NO
```

7.2.2 Compiling QD for higher precision arithmetic

MESMER uses numerical matrix techniques to formulate and solve the ME. Because of this, MESMER is not immune to numerical precision problems. In the Energy Grained Master Equation, the origin of these effects and when they occur is reasonably well understood, although solutions to these problems are less well understood. In general, numerical problems arise for very deep wells, very low temperatures, and very low pressures. MESMER includes a few different ways of dealing with numerical precision problems when they arise. The Reservoir state and contracted basis set approaches are elegant ways of manipulating the mathematical formulation of the ME to delay the onset of numerical problems; however, we have also written MESMER to incorporate a brute force technique for doing the same – carrying out arithmetic using significantly increased precision available in the so-called QD libraries written by Yozo Hida. To accommodate the increased precision libraries, MESMER may be built with different versions of QD. For the compilation of QD package please refer to http://crd.lbl.gov/~dhbailey/mpdist/. The QD installation steps are described in `INSTALL` file of the `/qd` folder.

Briefly, QD installation should require no more than the following three commands executed within the directory `Mesmer/qd/`:

```
chmod +x configure
./configure
make
```

When QD executes configure, it requires that certain environment variables are defined, and in our experience, the most common difficulties to installing QD concern the fact that these environment variables are not defined on a particular system. Executing

```
./configure --help
```

will show the list of important environmental variables required by QD. For example, QD may not recognize the appropriate C++ and Fortran compilers, and this is mostly like because the environment variables FC and CXX have no value. The user may check the value of the CXX and FC environment variables on their system by executing

```
echo $CXX; echo $FC
```

If the above commands return no value, and the respective C++ and Fortran compilers available on the system are g++ and gfortran, the environment variables may be set as follows:

```
CXX=g++; export CXX; FC=gfortran; export FC
```

In some cases where the user has no system administrator's privilege to install the library, they will need to ask the system administrator for help installing the QD package.

7.2.3 Compiling and Running the Main Executable

Following successful compilation of both the TinyXML and QD libraries, the main MESMER executable may be compiled. If the QD libraries have been built using files other than `qd_real.h` and `dd_real.h`, then the header file `MesmerPrecision.h` must be altered so that it refers to the correct filenames. For the standard QD install that we expect most users will utilize, changing `MesmerPrecision.h` will not be necessary. Tests that we have run on a variety of 32 and 64 bit LINUX architectures have shown that the MESMER executable may be unreliable if compiled with g++ compilers earlier than version 4.1.0, and with optimization flags greater than `-O2`. Thus, we recommend that users test their compiled code against the test suite detailed below.

If the user installs both TinyXML and QD themselves with no complicated changes to the standard install, then the MESMER `Makefile` in `/src` shouldn't need any alteration, as it is presently set up with the options that most users will require. To install the main executable, all that should be required is to go to the `/src` folder and do

```
make
```

or

```
make install
```

The latter command will copy the executable to the /bin folder after a successful compilation. By default, compilation will proceed with debug option flags applied, to compile with the full optimization flags then add the specification DEBUG=NO to the command line e.g.,

```
make install DEBUG=NO
```

If you need to recompile MESMER, you can also use the command

```
make remake
```

This will remove the previous built object files in /src folder and do a clean recompile/installation. This is useful when there is a clock skew between the local computer and remote cluster; hence all files on the remote cluster will be recompiled regardless of the time attributes on files. Note that if you don't do a make install, then the executable will reside in /src and won't be copied to /bin.

Should problems occur in compiling MESMER, or if you used more complicated ways of installing TinyXML and QD, we include a brief discussion of what must be in place to compile the Main MESMER executable. Additional guidance may be found within the comments in the Makefile itself. Compilation of the main MESMER executable requires linking with TinyXML and QD. To be sure this is done correctly, verify that the MESMER Makefile refers to the correct location of libqd.a and tinyxml.a within the LIBS field. If you successfully compiled the QD and Tinyxml packages, then paste something like this into its appropriate location with the MESMER Makefile:

```
LIBS:= ../tinyxml/tinyxml.a /usr/local/newqd/lib/libqd.a
```

where the paths of libqd.a and tinyxml.a may be relative or absolute.

Similarly, in the INCS section of the Makefile, specify the absolute or relative location of both the tinyxml/ and the /include folder.

```
INCS := -I../tinyxml/ -I/usr/local/qd/include
```

Where the /include folder specifies where qd_real.h and dd_real.h live. So long as the LIBS and INC sections of the Makefile are correct, then you should be able to carry on with compilation of the main MESMER executable using make or make install as discussed above.

7.2.4 Running on Linux/UNIX/Mac

Calling MESMER in UNIX/Linux/Mac systems is similar to Windows systems. The only difference is the name of the executable (`mesmer.exe` in Windows and `mesmer` in Linux). Assuming that you have done a `make install` so that the executable resides in `/bin`, then from the directory where `filename.xml` is located, one can call MESMER in UNIX/Linux systems by typing:

```
./~path/bin/mesmer filename.xml
```

where `~path` specifies the location of the `bin/` folder on your machine. If you would rather call MESMER by simply typing

```
mesmer filename.xml
```

without having to specify the executable path every time, then you have to export the directory in which the mesmer executable resides to the appropriate environment variables. One can see a list of the input options that may be specified with the MESMER executable by typing

```
-?
```

after the MESMER executable.

7.3 Testing MESMER on Windows and Linux/UNIX/Mac

Following MESMER compilation and/or installation, it is a good idea to perform some tests to ensure that your executable gives similar answers to the test jobs that we used during development. Tiny differences from those in the test jobs are probably OK, and likely arise from numerical issues due to OS/architecture combinations; however, larger errors are a cause for concern and you should check your compilation sequence. Under Windows, enter DOS command line mode, go to the `MesmerQA/` directory, and type the following command:

```
QA
```

This command executes a script called `QA.bat` which runs MESMER for each test system included within each file in the `MesmerQA` directory. Each system included in the test suite has a folder in which its input file is located. The input files for each test system are specified in section 11.1. Within the folder for each test system is another folder called

baselines/<platform>, into which the output from the QA.bat script is copied as test.test, where <platform> indicates the specific platform the test was run on e.g. Win32. The baselines/<platform> directory also includes a file called mesmer.test, which contains the output obtained by the developers for the corresponding test input file. The user needs to verify that the results generated by their executable in test.test are nearly identical to those in mesmer.test for each system. If the user carries out any MESMER development, then all changes to the code should be checked against the mesmer.test baselines. Additional instructions for operating the QA command in developer's mode are included within comments that are written in the QA.bat script.

For Linux/UNIX/Mac, an equivalent script is available and has the name QA.sh. Invocation of this script will depend on the command shell that is being used but is otherwise standard. The results can be examined in a similar way.

7.4 MESMER command line

All of MESMER's chemistry input and much of the program control is in the XML-formatted datafile described in the next section. The command line interface offers some options which mostly concern the location of files and only a few display/control tasks. The interface is the same in Windows and Linux. Typing

```
mesmer -?
```

will display the complete set of options for the command line. In normal execution of MESMER you would use:

```
mesmer infile.xml -o outfile.xml
```

If outfile.xml already existed, it would be renamed outfile_prev.xml, and any existing file of this name will be deleted. If you simply type

```
mesmer infile.xml
```

then output is written to mesmer_out.xml and there is the same single layer of buffering (i.e., renaming and deleting protocol) as with an explicitly named output file.

A useful command line option is the –N option, which when typed as

```
mesmer.out -N acetyl_O2_0003.xml
```

will prefix name all of the MESMER output files, *.test file, *.log file, and *.xml using the acetyl_O2_0003 prefix.

7.5 MESMER environment variables

The environment variable MESMER_AUTHOR can be set so that the user's name appears in the metadata section of the output XML file.

MESMER_DIR specifies the directory containing the executable, and is also where MESMER looks first for the files defaults.xml and librarymols.xml. If MESMER_DIR is not set, MESMER looks for these files in the directory two levels up from the current directory. This is appropriate when the current directory is one of the test system directories provided with the MESMER distribution.

On UNIX, Linux, and Mac platforms, MESMER_DIR can be used to simplify running MESMER from any directory, without having to specify the location of the MESMER executable every time:

```
mesmer filename.xml
```

To do this, set MESMER_DIR to the location of the MESMER executable by

```
MESMER_DIR="\user\username\Mesmer"
export MESMER_DIR
```

If you want the MESMER_DIR variable set every time you login to your machine, then add the above two lines to the relevant login scripts. The files defaults.xml and librarymols.xml also need to be in the same directory. If the user is running using the PBS command qsub, then the above two lines are usually are placed in the beginning of the qsub script file, which indicates that the commands are executed by the login shell every time the shell is initiated.

Under Windows, an entry in the PATH variable provides the location of mesmer.exe. If the Windows installer has been used, the MESMER_AUTHOR, MESMER_DIR, and PATH environment variables will have been set appropriately. MESMER_AUTHOR can be set temporarily, for the duration of a command window, by typing

```
set MESMER_AUTHOR name
```

8 MESMER data files

MESMER data files are in XML format and are intended to be more than a temporary means of transferring data to the program. They are more generally intended to be a representation of the chemical system – i.e., a set of reactions – which may (eventually) be used by other applications. Running MESMER produces an output file which is an augmented input file – it has all the original information together with additional data calculated by MESMER and the default values of parameters that were not explicitly specified. Consequently, any datafile can be used as input file.

The files can contain data in excess of that required by MESMER. For instance, they may contain chemical structure information, which is not used by MESMER, but helps to define the system unambiguously and can be used in the presentation of results. One of the reasons XML format was chosen because of the availability of tools and technologies for reformatting the data for presentation or reuse by other programs.

8.1 Editing and Viewing Data Files

With a very basic knowledge of XML syntax, any ordinary text editor may be used to read and edit a MESMER XML data file, but their construction and viewing is facilitated by the use of a specialised XML editor. Many commercial editors are available. Free ones for Windows include:

- Microsoft XML Notepad, which hides the syntax but emphasises the tree structure
- The editor in VS2010, which is good for syntax checking, but is part of a large development system – probably an excessive download unless you intend to do development.
- Notepad++, which has the basic capability of expanding and collapsing the XML tree structure.

MESMER data files can be viewed in a more user friendly way with Firefox 3, as shown in Figure 4.1. For Firefox to translate the XML data file, it requires with some XSL formatting

files – mesmerDiag.xsl and mesmer1.xsl – which need to be in a directory two levels above the XML data file and are usually in the MESMER root directory. In order to use your version of Firefox 3 to view the MESMER input files, you need to alter one of the Firefox defaults as follows:

- type `about:config` in the address bar. You will get a warning; receive it and carry on
- scroll down to `security.fileuri.strict_origin_policy`, right click on it, and change it to false;
- restart Firefox.

Now you can use it to view MESMER *.xml input files.

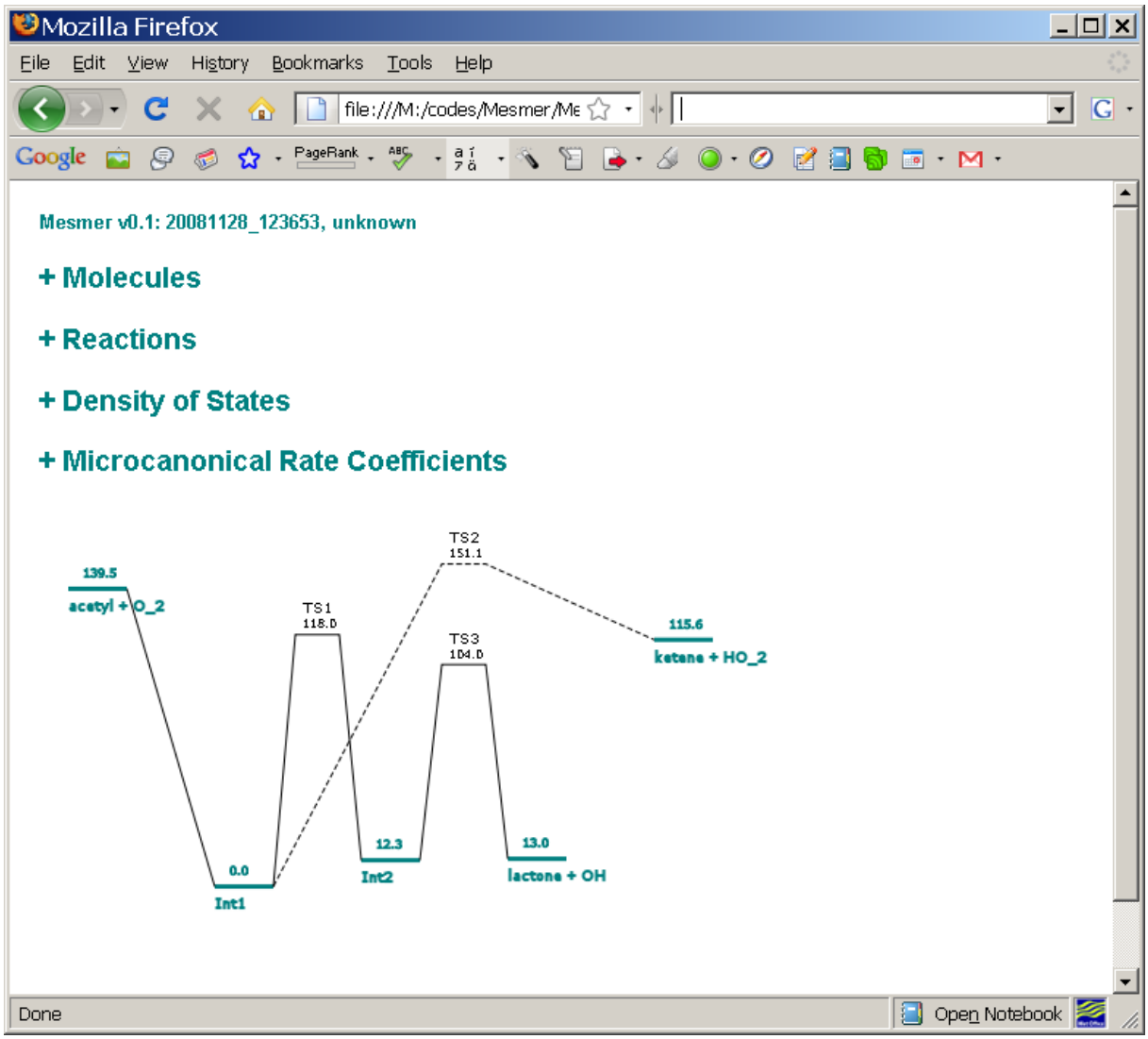

Figure 4.1 Viewing the potential energy surface of Acetyl+O_2 reaction by opening the XML input file in Firefox 3.

MESMER itself provides some assistance in constructing data files. If certain required items are not present, MESMER can insert them and prompt the user to check whether the inserted values are appropriate. See the tutorial "Constructing a Datafile from Gaussian output".

8.2 The basics of the *.xml input file

In the material that follows, we discuss the structure of MESMER input files. This discussion will be of significantly more benefit to the user if they use an XML editor to examine some of the sample input files included within the `MesmerQA` directory. Mesmer's input structure follows naturally from the XML data structure.

The MESMER data file has a top level element `<mesmer>`, and below it has the following sections, all of which have straightforward titles:

(1) `moleculeList` specifies the molecules relevant to the ME system, as well as associated properties of the molecules

(2) `reactionList` specifies the reactions relevant to the ME system to be modelled, and associated properties of the reactions

(3) `me:conditions` specifies the conditions (e.g., temperature, pressure, bath gas) under which a particular ME model is to be run

(4) `me:modelparameters` specifies the parameters relevant to the model (e.g., grain size and the maximum grain energy)

(5) `me:control` specifies program options concerning the content of the output file;

Both `moleculeList` and `reactionList`, which define the chemistry of the system, are based on CML (Chemical Markup Language) and this is the default namespace. Several programs work with CML; the tutorial mentioned in the previous section uses OpenBabel, to convert from other formats to CML. For more information about the CML schema can be found here

8.2.1 moleculeList

moleculeList is composed of each molecule involved in the ME system, and represents the distinct molecules or molecular configurations involved in the ME system. A molecule may represent one of the following: (1) an individual reactant in an association process, (2) an association product (i.e., an adduct), (3) an isomer, (4) a transition state, (5) an individual product from a dissociation reaction, or (6) a bath gas.

Each molecule includes the following:

(1) An id attribute, which is used to identify the molecule in other portions of the input file (e.g., the reactionList). An id of a molecule can be arbitrary as long as it is composed of ASCII characters, but it must be distinct from that of other molecules;

(2) An optional description attribute, which is available for the user to add their own comments regarding a particular molecule should they so choose;

(3) An atomarray, which is not necessary for the calculation to proceed, but which is part of the CML. The elements are each individual atom within the molecule. In the case that a molecule is no more than a single atom, as is often true for a reactant in an association reaction, a product in a dissociation reaction, or a bath gas, then atomarray has only a single constituent atom. Similar to molecule, each atom has the following attributes: an id, which is used to identify the atom in other portions of the input (e.g., the bondarray), and an elementType;

(4) A bondarray, whose elements are each bond between the constituent atoms in a molecule. Each bond has the following attributes: atomRefs2, which specifies atoms that are bonded (using the id attribute of atom) and order, which specifies the bond order. In the case that a molecule is no more than a single atom, it does not have a bondarray. It is important to note that a MESMER calculation requires neither an atomarray nor a bondarray for a successful calculation. However, these are features of the CML on which MESMER input syntax is based, and further applications may be available for interpreting these data structures.

(5) spinMultiplicity is an attribute on molecule;

(6) A propertyList array, which includes the following:

a. `me:DOSCMethod`, which specifies the method for MESMER to use in calculating rotational density of states. Classical and quantum mechanical methods for calculating the external rotational density of states are available, and they are specified with the text `ClassicalRotors` and `QMRotors`. Table 3.1 gives the classical and QM methods that are presently included in MESMER, and the manner in which the program recognizes the type of calculation to perform. Neither bath gas molecules nor products of sink reactions require calculation of the rotational state density and thus do not require a specification of `me:DOSCMethod`; however, specifying such a method will not affect program execution. (The criterion for a near symmetric top is that the modules of the asymmetric parameter, κ, should be within 5% of unity.)

Type of rotor	MESMER recognition criteria	Rotor type	DOS expression
not a rotor	no nonzero rotational constants available in the input	Classical	$\rho(E) = 0$
2d linear classical rotor	only one or two nonzero rotational constants provided in the input	Classical	$\rho(E) = \frac{1}{\sigma B}$
3d classical rotor	three rotational constants provided in the input	Classical	$\rho(E) = \frac{1}{\sigma}\sqrt{\frac{4E}{ABC}}$
Spherical top	$I_A = I_B = I_C$	QM	For $A = B = C$ $E_r(J,K)=BJ(J+1)$ $g_{JK} = (2J+1)^2$
Oblate or near oblate symmetric top	$I_A = I_B < I_C$	QM	For $A = B > C$ $E_r(J,K)=BJ(J+1)+(C-B)K^2$ $J=0,1,2;K=0,\pm1,\pm2,\ldots, \pm J$ $g_{JK} = 2J+1$
Prolate or near prolate symmetric top	$I_A < I_B = I_C$	QM	For $A > B = C$ $E_r(J,K)=BJ(J+1)+(A-B)K^2$ $J=0,1,2;K=0,\pm1,\pm2,\ldots, \pm J$ $g_{JK} = 2J+1$
Asymmetric top (King et al, *J.Chem.Phys.* **11**, 27 (1942))	$I_A < I_B < I_C$	QM	For $A > B > C$ $E_r(J)=(A+C)J(J+1)/2+\varepsilon(\kappa)$ $g_{JK} = 2J+1$

Table 4.1: methods for calculating DOS that are presently available in MESMER. Note that the rotational constants have the unit of cm^{-1}. g_{JK} are the degeneracies of (J,K) energy levels.

b. Extensions of the density of states calculation methods. Typically this through the specification of an extra density of state method via the tag

me:ExtraDOSCMethod. An important example in this respect is the extension that allows the calculation of hindered rotor densities of states. The hindered rotor facility is implemented as a plug in class and can be invoked using <me:ExtraDOSCMethod name="HinderedRotorQM1D">. The details of this class are given 63.

c. The me:reservoirSize flag is only relevant if the reservoir state approximation is being used (see below) and applies only to isomer wells, and specifies the reservoir size for the reservoir state approximation. The lower bound of the reservoir is always the well bottom, and the upper bound of the reservoir is specified by the user in energy units. If a negative number is supplied, then the reservoir upper bound is located that far (in energy) below the lowest available reaction threshold for a particular well – typically a few *kT*. If a positive number is supplied, MESMER will determine the upper bound that far (in energy) above the well bottom. MESMER will apply a correction if the user input specifies a reservoir upper bound that is higher than the lowest barrier. The energy may be input in any of several units, which must be specified in me:units. The user must determine whether the results of reservoir grain approximation are in agreement with the full master equation and whether it is an appropriate approximation. The syntax below specifies a reservoir spanning the well bottom to 2 kJ mol^{-1} beneath the lowest threshold.

```
<me:reservoirSize units="kJ/mol">-2.0</me:reservoirSize>
```

d. If the molecule being modelled is a minimum energy crossing point (MECP) – i.e., a transition state for spin forbidden crossing, then data is required. Both WKB and LZ corrections require specification of the following:

   i. A root mean squared spin orbit coupling element, specified following me:RMS_SOC_element with units="cm-1"

   ii. The norm of the vector representing the gradient difference at the MECP, specified following me:GradientDifferenceMagnitude with units="a.u./Bohr"

iii. The reduced mass for movement along the direction orthogonal to the crossing seam, specified following `me:GradientReducedMass` with `units="a.m.u."`

iv. WKB transmission probabilities require one additional input: the geometric mean of the norms of the gradients on the two surfaces at the MECP, specified using `me:AverageSlope` with `units="a.u./Bohr"`

e. `property` elements of a `molecule`. Each `property` element has a `dictRef` attribute which specifies what type of `property` is subsequently defined. A `property` type which requires a list, such as vibrational frequencies and rotational constants, is specified in an `array`, whereas one which requires only a single number, such as zero point energy location or a symmetry number, is specified to be `scalar`. For `property` types which have associated units, (e.g., vibrational frequencies, rotational constants, and zero point energy (ZPE)) the units are specified as an attribute of `array` or `scalar`. Unitless quantities such as spin multiplicity, symmetry number, or frequency scale factor, are specified to be `scalar`. Table 4.2 gives the values of `dictRef` (i.e., property attributes) currently recognized by MESMER, specifies whether they are `scalar` or `array`, and specifies the `units` presently available in MESMER.

f. The `me:ZPE` property defines the principal features of potential energy surface of the reaction system being modelled: it locates the stationary values of the potential energy surface relative to each other. It is important to note that the relative locations are defined in terms of the zero point energy of each species involved in the reaction system. Thus for an isomerisation reaction, the difference of the `me:ZPE` values of the two isomers corresponds to the heat of reaction at 0 K. Because MESMER is based on ZPE values, care must be taken when using data obtained from *ab initio* methods, that is classical potentials must be adjusted for ZPE and any corrections to vibrational frequencies (e,g, application of scaling factors to

correct for electron correlation effects etc.) used in these adjustments must be applied to frequencies supplied to MESMER, either directly or via the property me:frequenciesScaleFactor. In addition there are two important points that need to be accounted for when specifying zero point energy:

i. While different units are available for input of a molecule's ZPE, the selected units must be consistent for every molecule in the input file with respect to some arbitrary reference energy.

ii. For association and dissociation reactions, the respective sum of the reactant and product ZPEs for each molecule must equal the total energy of the products or reactants. That is, if the ZPE of A+B is equal to *x* kJ/mol for a particular system, there are infinite number of ways in which this may be specified so long as the ZPE of A added to the ZPE of B is equal to *x* kJ/mol. It is usually mostly convenient to specify the excess reactant a having a ZPE of zero and ascribe the ZPE difference to the deficient (pseudo-isomer) species.

g. As an alternative to ZPE, energies can be input as Heat of Formation at 0K, either with a conventional basis (me:Hf0) or with ground-state atoms, rather than standard states of elements assigned a zero value (me:HfAT0). Heat of Formation at 298K (me:Hf298) is also supported. When the XML is parsed, these are converted and a property with dictRef = me:ZPE written to the output XML file with an attribute describing the origin of the data. When this XML file is used for input, the ZPE data is used, because the priority order for defining the energy is me:ZPE, me:Hf0, me:HfAT0, me:Hf298 .

dictRef value	scalar or array	Available input units
me:ZPE	Scalar	kJ/mol or kJ per mol cm-1 or wavenumber kcal/mol or kcal per mol Hartree or au
me:Hf0	Scalar	as above
me:HfAT0	Scalar	as above
Me:Hf298	Scalar	as above
me:rotConsts	Array	cm-1
me:symmetryNumber	Scalar	unitless, no units specification
me:frequenciesScaleFactor	Scalar	unitless, no units specification
me:vibFreqs	Array	cm-1
me:MW	Scalar	Amu
me:spinMultiplicity	Scalar	unitless, no units specification A spinMultiplicity attribute on <molecule> is a preferred alternative.
me:epsilon	Scalar	K, no units specification
me:sigma	Scalar	Angstroms, no units specification
me:deltaEDown	scalar	cm-1

Table 4.2: Some of the values of dictRef recognized by MESMER, whether the associated input is scalar or an array, and the available units for the input values.

8.2.2 reactionList

reactionList is composed of each individual reaction that may occur in a ME system. Effectively, the reactionList specifies which molecules are connected via which transition state. A reaction may be one of the following: (1) an isomerization, (2) an association, or (3) a dissociation.

Each reaction may include the following:

(1) An id attribute, which is used to identify the reaction;

(2) An active attribute. If this is false the reaction is ignored by MESMER. This feature makes it possible to do calculations on only part of a complex reaction system. A molecule can also have this attribute, but this does not affect the calculation. Inactive reactions and molecules are shown either greyed or omitted (at the users choice) when the XML file is viewed in Firefox.

(3) One or more of a reactant and one or more of a product. An isomerization reaction (A→B) has one reactant and one product, an association reaction (A+B→C) has two reactants and one product, a dissociation reaction (A→B+C) has one reactant and two products, irreversible unimolecular

reactions (A→C) have one reactant and one or two products, and irreversible exchange reactions (A+B→C+D) have two reactants and two products. reactions may or may not have a me:transitionState, which is discussed further below. All reactants, products, and me:transitionStates are molecules, and have:

a. A ref, which should be identical to the molecule id specified in moleculeList, and

b. A me:type, which specifies the molecule's role in the ME model of the system. The possible values of me:type are given in Table 3.3.

(4) If me:type is excessReactant, then reaction requires a value of the excess reactant number density associated with me:excessReactantConc (in units of molecules cm^{-3});

(5) A me:tunnelling specification, which indicates whether and what sort of tunnelling corrections are to be implemented for a particular reaction. Presently, there are two types of tunnelling correction that have been implemented: a one dimensional tunnelling through an asymmetric Eckart barrier, using the method described by Miller,[1] and specified in MESMER with Eckart and a one dimensional tunnelling through a user defined potential using the semi-classical method described by Garrett and Truhlar, *J . Chem. Phys.*, 1979(83), 2921-2925, specified with WKB. More detail can be found in the section on plug-in classes.

(6) A me:crossing specification for invoking spin forbidden corrections to RRKM theory. Presently, there are two methods available for calculating spin forbidden transmission coefficients: (1) a Landau Zener method specified with LZ, and (2) a WKB method specified with WKB. Unlike the former method, the latter includes tunnelling corrections below threshold. Both methods are described by Harvey and Aschi, *Faraday Discuss.*, 2003(124), 129-143.

(7) A me:MCRCMethod specification, which indicates how microcanonical flux through the transition state is to be treated for a particular reaction. Presently, there are two methods for treating the transition state: (1) If me:transitionState is specified, then the SimpleRRKM method, which uses the standard, well known

RRKM expression to calculate $k(E)$, may be used. (2) If no transition state is specified, then `MesmerILT` may be used. This specifies that $k(E)$ are calculated using an inverse Laplace transform of $k^{\infty}(T)$s, which may be experimentally measured or theoretically calculated. This method is often convenient for reactions that do not have a well-defined energy barrier.

If `MesmerILT` is specified, then `reaction` requires three parameters from an Arrhenius fit to $k^{\infty}(T)$. The expression used by MESMER is:

$$k^{\infty}(T) = A^{\infty}\left(\frac{T}{T^{\infty}}\right)^{n^{\infty}} \exp(-E^{\infty}/RT)$$

where A^{∞} is the pre-exponential factor (`me:preExponential`), E^{∞} is the activation energy (`me:activationEnergy`) in units identical to those of the corresponding stationary point energies, and n^{∞} accounts for curvature in the observed $k^{\infty}(T)$ (`me:nInfinity`). (Note these parameters can also be floated during a fitting exercise.)

`me:type` value	MESMER definition
`deficientReactant`	Deficient reactant in an association or exchange reaction; modelled using a pseudo first order bimolecular source term and assumed to be thermalized in the ME model. Presently, reversible association reactions and irreversible exchange reactions are available
`excessReactant`	Excess reactant in an association reaction; its large concentration relative to the deficient reactant allows a pseudo first order treatment of the association step
`modelled`	Molecule that may undergo relaxation via collisions with the bath gas
`transitionState`	Molecular configuration that represents the phase space flux bottleneck for a particular reaction
`sink`	product molecule that acts as an infinite sink (i.e., an irreversible loss) from a particular well modelled in the ME

Table 4.3: different values of me:type and their corresponding definitions in MESMER

8.2.3 me:conditions

The tree structure for `me:conditions` includes the following elements:

(1) A specification of the `me:bathGas`, which is identical to the `id` attribute associated with the bath gas `molecule` in the `moleculeList`. The syntax should look something like:

```
<me:bathGas>He</me:bathGas>
```

(2) me:PTs, which is used to specifies the physical conditions at which the ME model is to be run. me:PTs consists of number of me:PTpair elements, which specify, as attributes, the pressure (me:P) and temperature (me:T), at particular master equation model is to be run. The number of me:PTpairs tells MESMER how many ME simulations to run. For example, three me:PTpairs would result in ME runs at three specified temperature and pressure pairs. Temperature is input in Kelvin, but the pressure may be input in any of several units, which must be specified in me:units. The units recognized by MESMER for pressure input include: (1) number density, which may be specified as particles per cubic centimeter, number density, or PPCC; (2) torr, specified as Torr or mmHg; (3) millibar, specified as mbar, (4) atmospheres, specified as atm, and (5) pounds per square inch, specified as psi. In addition to these attributes, the me:precision attribute determines the precision at which the calculation is to be run. More detail on how to use this element to specify experimental data, to be used for fitting, can be found in section 9.2.

(3) me:InitalPopulation, which specifies the initial population for each species in order to get the time evolution of the system. If there is a source term (i.e., an association reaction), MESMER will automatically set its initial population to 1.0. If there is more than one source term, MESMER will set the population of the first source term it encountered to 1.0, leaving the populations of the other source terms nil. If there is no source term in the system, MESMER will set the population of the first isomer it encountered to 1.0. Otherwise, the user can specify the initial population of the species by using this element, and the syntax should look like:

```
<me:InitalPopulation>
  <molecule ref="cyclopropene" me:population="1.0" />
</me:InitalPopulation>
```

8.2.4 me:modelParameters

The tree structure for me:modelParameters includes the following elements:

(1) me:grainSize includes a specification of the grain size to be used in partitioning the system phase space. It has an associated value and associated units. The units available are identical to those given in Table 2.2 for specification of ZPE. For a

convergent solution of the ME utilizing the exponential down model, the grain size should be smaller than the $\langle \Delta E_d \rangle$, the average energy transferred in a deactivating collision, values of any of the `modelled` molecules.

(2) `me:energyAboveTheTopHill` specifies the energy range to be spanned by the grains, where the units are *kT*. The value of this parameter sets up a model wherein the energy grains span a region possessing *x kT* of energy in excess of the highest energy stationary point. In general, the maximum grain energy in the ME should be at least 20 *kT* above the highest energy molecular configuration, including reactants. If the maximum grain energy is too large, then numerical errors may result given that the probability for activating collisions is so small as to exceed machine precision.

(3) `me:maxTemperature` applies if more than one `me:PTpair` has been specified in `me:conditions`. If `me:maxTemperature` is not specified, then each ME run at a particular `me:PTpair` will use the `me:energyAboveTheTopHill` value to determine the maximum grain energy – i.e., the maximum grain energy will be different at different temperatures. Using `me:maxTemperature` makes the maximum grain energy at each `me:PTpair` identical, choosing the maximum grain energy to be *x kT* above the highest energy stationary point, where *T* is equal to the value set in `me:maxTemperature` regardless of the temperature specified in the `me:PTpair`.

8.2.5 me:control

The tree structure for `me:control` includes several elements, all of which determine the content of the output file and may be turned off and on by commenting or un-commenting any of the following items:

- `me:calculateRateCoefficientsOnly`, which makes MESMER only calculate TST rate coefficients without doing the diagonalization required by a full ME treatment.
- , which prints the eigenvalues obtained from diagonalization of the full system collision matrix. The integer included within this element specifies how many eigenvalues to truncate in the printing. If *ngrn* is the number of grains in the

collision operator, and the value in me:eigenvalues is *x*, then *ngrn* – *x* eigenvalues will be printed. *x* = 0 indicates that all of the eigenvalues should be printed.

- me:calcMethod, which specifies the type of calculation to be done. These calculations are implemented as plug-in classes (see plug-in class section for further details). For example, to explore the sensitivity of the computed results to a barrier height or an energy transfer parameter like $\langle \Delta E \rangle_d$ a grid search calculation might be performed and this is specified as **<me:calcMethod> gridSearch </me:calcMethod >** in the **<me:control></me:control>** section.. The present type of allowed calculations are:

    a. simpleCalc: the default, and does a normal set of ME calculations at each of the specified pressure and temperature points.

    b. gridSearch: initiates a grid search over parameters specified by the user. χ^2 is calculated against experimental rate data, for every point in parameter space specified by the user.

    c. fitting: fits experimental rate data by optimizing parameters specified by the user, using a conjugate direction method based on the Powell algorithm.

    d. marquardt: fits experimental rate data by optimizing parameters specified by the user, using an implementation based on the Marquardt algorithm.

    e. ThermodynamicTable: Calculates thermodynamic functions for the species defined in the molecular list.

- me:printCellDOS, which prints out the cell DOS for the wells in the ME system

- me:printCellTransitionStateFlux, which prints out cell transition state fluxes.

- me:printReactionOperatorColumnSums, which prints column sums of the full, normalized collision operator.

- me:printGrainBoltzmann, which prints out the normalized equilibrium grain population of all wells in the ME system.
- me:printGrainDOS, which prints out the grain DOS for the wells in the ME system
- me:printGrainkbE, which prints *k(E)*s for backward reactions specified in the reactionList.
- me:printGrainkfE, which prints *k(E)*s for forward reactions specified in the reactionList.
- me:printTSsos, which prints the sum of states, *N(E)* at the TS for reactions specified in reactionList. In the case of ILT, unlike statndard RRKM methods, there is no explicit TS, and reaction flux is calculated directly. Nevertheless, an 'effective' TS sum of states may be calculated from the calculated reaction flux.
- me:printGrainedSpeciesProfile, which prints out time evolution of every grain in the logarithmic time scale.
- me:printGrainTransitionStateFlux, which prints out grained transition state fluxes.
- me:printReactionOperatorSize, which prints out the reaction operator. The text within this element specifies how many lower-right rows/columns to generate for printing. For example, if one specifies 20, then the lower-right square matrix block containing 20 rows/columns will be printed. In addition, one can specify negative numbers: -1 for printing whole matrix, -2 for printing lower-right square matrix with ½ the number rows/columns compared to the whole reaction matrix, -3 for printing lower-right square matrix with 1/3 the number rows/columns compared to the whole reaction matrix.
- me:printSpeciesProfile which prints out time evolution of every species in the logarithmic time scale.
- me:printTunnelingCoefficients, which prints tunnelling coefficients for those reactions where tunnelling corrections are specified.

- me:printTunnellingCoefficients, same as the previous one.
- me:printCrossingCoefficients, which prints transmission coefficients for those reactions where spin forbidden RRKM theory is requested
- me:shortestTimeOfInterest, which puts a user defined lower boundary of integration time for species profile. Usually this is helpful when the starting reaction is fast (greater than 10^{10} s^{-1}). User should put syntax as the following

  <me:shortestTimeOfInterest>10e-14</me:shortestTimeOfInterest>

  to specify the time boundary.
- me:MaximumEvolutionTime, which puts a user defined upper boundary of integration time for species profile.
- me:testDOS, which calculates and prints partition functions for the wells in the ME system using the grain DOS, cell DOS, and analytical forms.
- me:testMicroRates, which computes canonical rate coefficients at a range of temperatures using the grain DOS and the grain *k(E)*s.
- me:testRateConstants, which computes TST *k*(*T*)s for each reaction in the system.
- me:useTheSameCellNumberForAllConditions, which forces MESMER to use the same cell number (the highest cell number amongst all simulations) for all calculations. This option only applies for input files specifying multiple ME calculations.

  The following elements control the display of the energy level diagram generated from the XML data.
- me:hideInactive, which removes molecules or reactions with the attribute active="false" from the diagram. There is a control on the diagram which allows this to be toggled.
- me:diagramEnergyOffset which adjusts the displayed energy values of species. For example,

  <me:diagramEnergyOffset>0</me:diagramEnergyOffset>

makes the lowest energy species have a displayed energy of 0. This is almost essential when the energies are used directly from a computational chemistry program, because of the large offset that they have.

8.3 Summary Table: Molecular input variables in MESMER

The following table lists essential variables for molecules in the MESMER calculation. The different types of molecules in the table are bath gas (Bath), transition states (TS), Excess Reactants (ExcS), deficient reactants (DefS), sink molecules (Sink), and modelled molecules (MM). The variables in grey boxes are not input by user, but calculated by MESMER according to the input parameters.

Parameter	Bath	TS	ExcS	DefS	Sink	MM
atomic/molecular mass	•	•	•	•	•	•
σ (sigma)	•					•
ε (epsilon)	•					•
B_{xyz} (rotational contants)		•	•	•	•	•
σ_{sym} (rotational symm num)		•	•	•	•	•
ZPE (zero point energy)		•	•	•	•	•
ν_f (vibrational frequencies)		•	•	•	•	•
Spin multiplicity		•	•	•	•	•
Electronic Excitation		•	•	•	•	•
Number of cells		•only for RRKM	•	•	•	•
Number of grains			•	•	•	•
Energy of grains			•	•	•	•
Initial Population				•	•	•
Equilibrium fraction				•	•	•
$\langle \Delta E \rangle_d$ (delta E down)						•
Collision frequency						•
Collision operator size						•
Initial grain populations						•

9 Additional facilities and examples

Below, we introduce the structure of some basic XML, and provide some examples for utilizing MESMER functionality that was not discussed above. The best place to look for examples is in MesmerQA folder, the components of which are discussed in detail below. The examples inside the MesmerQA folder cover *most* of the functionality available in MESMER, but not *all* of it. As MESMER grows, so will the material that follows.

9.1 Basic XML Structure

A basic one atomic molecule should look like this in MESMER:

```
<molecule id="He">
  <propertyList>
    <property dictRef="me:epsilon">
      <scalar>10.2</scalar>
    </property>
    <property dictRef="me:sigma">
      <scalar>2.55</scalar>
    </property>
    <property dictRef="me:MW">
      <scalar units="amu">4.0</scalar>
    </property>
  </propertyList>
</molecule>
```

From this view one can see that all property elements are enclosed inside `propertyList` element, where the `<atomarray></atomarray>` element and `<bondarray></bondarray>` elements are not shown in this example. It is not necessary to include the `<atomarray></atomarray>` and `<bondarray></bondarray>` elements in a molecule, but in the future, MESMER may use the elemental data to check the credibility of the vibrational/rotational data provided by user, or even draw pictures of the molecular geometries on the PES.

Any complete XML element should be one of the following

```
<elementName parameterName="id"></elementName>
```

or

```
<elementName parameterName="id" />
```

where in the first expression, child elements can be inserted between two element bodies as

```
<elementName parameterName="id">
     <childElementName parameterName="id"></childElementName>
     <childElementName parameterName="id"></childElementName>
</elementName>
```

Be careful when converting the first expression to the second expression if you are editing on a text editor, especially on the location of the forward slash in the expression.

9.2 Comparing MESMER rate data to experimental values

It is often the case that values calculated by MESMER need to be compared with an experimental value, particularly during a fitting exercise. MESMER allows comparison with three types of experimental data: rate coefficients, yields and eigenvalues. These data can be specified for a give phenomenological reaction or species. The specification of this data is as child elements of the `me:PTpair` element described in section 8.2.3, and any number and type of data can be specified for a given set of conditions. During a fitting exercise all values are combined to give a total value χ^2 which is then optimized. The details of each type are as follows:

9.2.1 Experimental Rate Coefficients

Experimental rate coefficients in MESMER are defined by two different references, ref1 and ref2, where ref1 is the reactant, and ref2 is the product. It is easy to identify ref1 and ref2 for any reaction in MESMER by running a preliminary calculation, and inspecting the output in the Bartis Widom phenomenological rate coefficient analysis in mesmer.test. All the rate coefficients are output in the format ref1 -> ref2, unless the rate coefficient is that for phenomenological loss, in which case it is specified as a loss reaction. For a loss reaction, ref1 and ref2 are identical, as in the example shown below, where we compared the experimentally measured rate coefficient for acetyl loss to the phenomenological rate coefficient calculated by MESMER. To get MESMER to output the square of the difference between a calculated and an experimental rate coefficient divided by the square of the error (i.e χ^2), enter the following:

```
<me:PTpair me:units="Torr" me:P="200.12" me:T="500" me:precision="double" >
  <me:experimentalRate ref1="acetyl" ref2=" acetyl " error="6750">78000</me:experimentalRate>
</me:PTpair>
```

where the experimental rate goes in between the ><, as discussed above. If there are no experimental rates to compare with, the child element can be left undefined:

```
<me:PTpair me:units="Torr" me:P="200.12" me:T="500" me:precision="double"></me:PTpair>
```

or even shorter

```
<me:PTpair me:units="Torr" me:P="200.12" me:T="500" me:precision="double" />
```

Since double is the default precision in MESMER, one can write

```
<me:PTpair me:units="Torr" me:P="200.12" me:T="500" />
```

instead.

9.2.2 Experimental Yields

Experimental yields can be specified in the following way:

```
<me:PTpair me:units="PPCC" me:P="4.46E18" me:T="195" me:precision="dd">
  <me:experimentalYield ref="OH-1" error="0.0050" yieldTime="1.e-03">0.0509
</me:experimentalYield>
</me:PTpair>
```

The `ref` attribute specifies the species for which the yield is to be determined, in the above example, the species species "`OH-1`". It is often the case that yields are measured at specific time after reaction has been initiated, and the `yieldTime` attribute allows this time to be specified. If this attribute is not defined then the infinite time (or, in the case of a conservative model, equilibrium) yield is calculated.

9.2.3 Experimental Eigenvalues

In some cases the relaxation of the system is the experimentally observed quantity and it is often best to compare the observed relaxation time constant with the eigenvalues generated by MESMER. Such data can be specified as follows:

```
<me:PTpair me:units="PPCC" me:P="7.6E18" me:T="440." me:precision="dd">
  <me:experimentalEigenvalue EigenvalueID="2" error="200.">14000.</me:experimentalEigenvalue>
</me:PTpair>
```

MESMER typically generates a large number of eigenvalues so it is important to specify which eigenvalue is to be used for comparison and the attribute `EigenvalueID` allows this association to be made. The eigenvalues are labelled from 1 to *N*, where *N* is the size of the collision operator, and the eigenvalues increase in magnitude from 1 to *N*. It is usually the case that only the first few eigenvalues are of chemical significance.

9.3 Specifying Numerical Precision

As discussed above, double is the default MESMER precision; however, quad-double and double-double precisions can also be specified inside **<me:PTpair>** element, the syntax is

```
me:precision="quad-double"
me:precision="double-double"
```

or

```
me:precision="qd"
me:precision="dd"
```

9.4 Specifying Parameter Bounds and Constraints

During a fitting exercise or a simple grid search the parameters that are to be varied need to be specified. For example, one might want to float exponential down parameter $\langle \Delta E \rangle_d$ and the Arrhenius parameters in a fitting exercise. To indicate that a parameter is variable, the user must add additional attributes that specify the search range and the step size. For example, an Arrhenius parameter A^{∞} might be declared as variable by the following statement:

```
<me:preExponential lower="4.6e-12" upper="6.201e-12" stepsize="2.0e-13">6.00e-12</me:preExponential>
```

In this expression, there are three additional parameters for the `me:preExponential` element: the lower bound (with value 4.6×10^{-12}), upper bound (with value 6.2×10^{-12}), and step size (with value 6.0×10^{-13}). All three additional parameters are used in a grid search. For either type of fitting, the step size parameter is ignored.The order of these three elements is not important. According to XML formatting protocol, the number `6.00e-12` within the element pair of `me:preExponential` must be supplied, this number will be ignored in the case of a grid search, but will be taken as the starting point for either of the fitting methods (in addition, for ZPE parameters, it is actually used to display the PES using Firefox).

The parameters that can be varied, and their associated XML element, are given in the following table:

Parameter	Description	XML Element
A^{∞}	Arrhenius pre-exponential factor	me:preExponential
E^{∞}	Arrhenius activation energy	me:activationEnergy
n^{∞}	Modified Arrhenius parameter	me:nInfinity
ΔH_0^0	Zero-point energy of potential energy surface stationary point features.	me:ZPE
$\langle \Delta E \rangle_{d,ref}$	Exponential down energy transfer parameter.	me:deltaEDown
n	Exponential down temperature exponent: $\langle \Delta E \rangle_d = \langle \Delta E \rangle_{d,ref} \left(\frac{T}{T_{ref}} \right)^n$ where T_{ref} is 298.0 K by default but can be set via the attribute referenceTemperature.	me:deltaEDownTExponent
ν_i	Imaginary frequency used in determining tunnelling coefficients.	me:imFreqs

In complex systems, there may be a very larger number of parameters that can be varied. This can make fitting exercises very expensive. However, it is often the case that some parameters are, or can be approximately related. An example is the $\langle \Delta E \rangle_d$ parameters for two isomers that are similar in structure and which, to a first approximation, can be regarded as the same. Such parameters can be varied in a constrained way by a sequence similar to the following:

```
.
.
.
<property dictRef="me:deltaEDown">
   <scalar label="IM1" units="cm-1" lower="100" upper="400" stepsize="10">174</scalar>
</property>
.
.
.
<property dictRef="me:deltaEDown">
   <scalar label="IM2">161</scalar>
</property>
.
.
.
<me:control>
   <me:calcMethod>marquardt</me:calcMethod>
   <me:MarquardtIterations>9</me:MarquardtIterations>
   <me:MarquardtTolerance>1e-6</me:MarquardtTolerance>
   <me:MarquardtDerivDelta>0.025</me:MarquardtDerivDelta>
```

```
    <me:ParameterConstraints>
      <me:Constraint me:Independent="IM1" me:Dependent="IM2" me:ConstraintFactor="1.0"
me:ConstraintAddand="0.0" />
    </me:ParameterConstraints>
  </me:control>
```

In the first part of this sequence a $\langle \Delta E \rangle_d$ parameter is defined in the usual way, but with the additional definition of a label which will be used in the subsequent constraint definition. In the second part of the sequence another $\langle \Delta E \rangle_d$ parameter is defined with a different label, but this time without any range attributes. The third section shows the control sequence of the job in which the constraint is defined using the two previously defined labels. The other attributes in the constraint definition determine the nature of the linear relation between parameters, i.e.:

```
me:Independent = me:ConstraintFactor * me:Dependent + me:ConstraintAddand
```

The default value for `me:ConstraintFactor` is 1.0 and for `me:ConstraintAddand` is 0.0, and here is no limit to the number of constraints that can applied.

9.5 Unimolecular and Reverse ILTs

As indicated above, MESMER includes two different ways for calculating the microcanonical rate coefficients – through the use of the RRKM expression, or the use of the ILT technique. The ILT technique is particularly useful when there is no easily identifiable energetic barrier, such as often occurs in the case of radical-radical reactions or non-adiabatic reactions. ILT offers a mathematical formalism for deriving *k*(E)s from an Arrhenius fit to a set of *k*(T)s, which may be obtained from experiment or theory. There are three possible uses of the ILT in MESMER: (1) for association reactions, where the Arrhenius expression is for the association reaction of one deficient reactant combining with one excess reactant to make a modelled molecule; (2) for isomerization reactions of one modelled molecule to another, where the Arrhenius expression is for the forward reaction; (3) for an irreversible reaction of one modelled molecule to one or two sink molecules, where the Arrhenius expression is for the forward reaction, and (4) for dissociation reactions of one modelled molecule to two sink molecules, where the Arrhenius expression is for the association of the two sink molecules. Implementation (1), which is the most common, we call the standard ILT. Implementations (2) and (3) are the unimolecular ILT, and (3) is the reverse ILT. The test file for the acetyl + O_2 reaction has examples of the standard ILT, so we do not discuss this here. The unimolecular ILT is identical to the standard ILT, except that the reaction replaces the two

reactants with a single modelled molecule. Below, we provide syntax for the reverse ILT, which in this case corresponds to the dissociation of I_2O_2 to OIO + I:

```
<reaction id="R2">
  <reactant>
   <molecule ref="IO_IO" me:type="modelled" />
  </reactant>
  <product>
   <molecule ref="OIO" me:type="sink" />
  </product>
  <product>
   <molecule ref="I" me:type="sink" />
  </product>
  <me:MCRCMethod>MesmerILT</me:MCRCMethod>
  <me:preExponential>1.4E-10</me:preExponential>
  <me:activationEnergy units="kJ/mol" reverse="true">0.0</me:activationEnergy>
  <me:TInfinity>298.0</me:TInfinity>
  <me:nInfinity>0.01</me:nInfinity>
</reaction>
```

The reverse ILT is requested with reverse="true" on the me:activationEnergy element, where the activation energy is referenced to the ZPE of OIO + I The input Arrhenius data is for the OIO + I association reaction, but reverse="true" tells MESMER to use that data for calculating $k(E)$s of the forward dissociation reaction – i.e., $I_2O_2 \rightarrow$ OIO + I. MESMER treats a typical dissociation reaction as irreversible – i.e., data for sink molecules is not required by MESMER (although names and ZPEs *are* required if you want a nice diagram in Firefox, and it's good for future bookkeeping of reaction data). However, for a reverse ILT, product data of the sink molecules is essential, and the MESMER will terminate if it is not provided. Finally, due to the mathematics of the ILT, it is necessary to point out that there are some constraints on: for the standard ILT, n^{∞} must be greater than -1.5, while for the unimolecular and reverse ILT, n^{∞} must be greater than zero. If a zero n^{∞} is desired, setting it to a very small number will introduce minimal error.

9.6 Secondary input files

Another facility for making use of a library of molecules is to use *secondary input files*. These are XML files specified on the command line after the main XML file. For instance, in

```
mesmer maininput.xml sec1.xml sec2.xml -o outfile.xml
```

`sec1.xml` and `sec2.xml` are secondary input files. MESMER inserts their contents into the main XML. So if the secondary file contained:

```
<moleculeList>
   <molecule id="mol1"> … </molecule>
```

```
    <molecule id="mol2"> … </molecule>
  </moleculeList>
```

the molecules would be inserted into the main `<moleculeList>`. Or a secondary file like:

```
  <me:control>
    <me:printSpeciesProfile />
  </me:control>
```

might be used to obtain a different output, without editing the main data file.

10 MESMER files explained

This section provides an explanation of important features of the more significant files in the source folder and produced during a calculation.

10.1 MESMER output files

Each MESMER calculation creates three output files. They are *.test, *.log and *.xml;

10.1.1 mesmer.test

Much of the mesmer.test file contains information that user chooses to print out in <me:control>. Here we review some of the most important items in this file and provide brief explanations.

10.1.1.1 Partition Functions and State Densities

```
Test rovibronic density of states for: <molecule name>
```

This line is followed by columns like the following

```
      T            qtot            sumc            sumg
    200   4.38418e+006    4.4283e+006   4.40474e+006
    300   5.15155e+007   5.21956e+007    5.2071e+007
    400   5.69583e+008   5.78432e+008   5.77654e+008
    500   6.05474e+009   6.15825e+009   6.15289e+009
    600   6.12019e+010   6.17397e+010   6.16902e+010
```

which gives the canonical partition function of the named molecule. Column `qtot` is obtained through analytical approximation of the partition function while columns labelled `sumc` and `sumg` calculate the partition functions from summing the state averaged cell and grain densities, respectively.

```
Grain rovibronic density of states of <molecule name>
```

This section prints rovibronic state densities for the molecule in grains, where the first column is the mean grain energy in cm^{-1}.

10.1.1.2 k(*E*)s & Tunnelling Corrections

```
k_f(e) grains for <reaction name>
```

This section gives the forward microcanonical rates calculated from the grains.

```
k_b(e) grains for <reaction name>
```

This section gives the reverse microcanonical rates calculated from the grains.

```
Tunnelling coefficients for: <reaction name>

V0 = 11201.4, V1 = 10047.3, barrier0 = 9864, barrier1 = 8834, imFreq = 5.0605e+13
```

This section gives tunnelling coefficients of the reaction. V0 and V1 are forward and reverse classical barrier height of the reaction. Barrier0 and barrier1 are zero-point energy differences between the transition state and the respective reactant and product, with units in cm^{-1}. The data used to calculate transmission coefficients for spin forbidden RRKM theory is printed out in the *.log file.

10.1.1.3 Equilibrium Fractions

```
Eq fraction matrix:
{
 -4.9732e+013             1             0
            0   -0.0036887             1
            1             1             1
}
inverse of Eq fraction matrix:
{
 -2.0108e-014 -2.0034e-014  2.0034e-014
  2.0034e-014     -0.99632      0.99632
  7.3899e-017      0.99632    0.0036751
}
Equilibrium Fraction for Int1 = 0.996325
Equilibrium Fraction for Int2 = 0.00367513
Equilibrium Fraction for acetyl = 2.00339e-014
```

This section uses rovibronic partition functions of isomers and pseudo-isomers to calculate equilibrium constants. Consider a three well system: e.g., A ⇔ B ⇔ C where the equilibrium constant for A ⇔ B is given by, $k_{eq} = k_1 = \frac{Q_B^{rve}}{Q_A^{rve}}$ and that for B ⇔ C is given by $k_{eq} = k_2 = \frac{Q_C^{rve}}{Q_B^{rve}}$. Therefore, the relation is defined by the following three linear equations:

$$-k_1 A + B = 0$$

$$-k_2 B + C = 0$$

$$A + B + C = 1$$

Equation 4.1

which is a 3 by 3 matrix and a vector which satisfy the following:

$$\begin{pmatrix} -k_1 & 1 & 0 \\ 0 & -k_2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Equation 4.2

The equilibrium fraction of each isomer (or pseudo isomer, in the case of a source term) may be obtained by inverting the matrix shown above, and taking the elements in the final column of the inverse. Any system, with an arbitrary number of wells and connections, may be described by such a Matrix. Irreversible channels are not included within the calculation of the equilibrium fractions.

10.1.1.4 Eigenvalues

```
Total number of eigenvalues = 347
Eigenvalues
```

This section gives the eigenvalues of the reaction (system collision) operator, all of which should be less than or equal to zero. Within the energy grained Master Equation, collisional energy transfer in the grained phase space and inter-conversion between species is described using a set of coupled differential equations:

$$\frac{d}{dt}\boldsymbol{p} = \boldsymbol{M}\boldsymbol{p}$$

Equation 4.3

where $\boldsymbol{p}$ is the population vector containing the populations of the energy grains for all isomers and pseudo-isomers, and $\boldsymbol{M}$ is the matrix that determines population evolution due to collisional energy transfer and reaction. The discretized matrix $\boldsymbol{M}$ is diagonalized, and the eigenpairs give a solution of the form:

$$\boldsymbol{p}(t) = \boldsymbol{U}e^{\boldsymbol{\Lambda}t}\boldsymbol{U}^{-1}\boldsymbol{p}(0)$$

Equation 4.4

where $\boldsymbol{p}(0)$ contains the initial conditions (i.e., $t = 0$) for each grain (i.e., $n_{iE}(0)$), $\boldsymbol{U}$ is matrix of eigenvectors obtained from diagonalization of $\boldsymbol{M}$, and $\boldsymbol{\Lambda}$ are the corresponding eigenvalues. For many chemical systems, one observes a separation in eigenvalues, with the bulk corresponding to very fast decay (i.e, a large absolute value), and a select few corresponding to slower decay (i.e, a small absolute value). The eigenvalues for fast decay correspond to the time scales for relaxation of activated intermediates and are often referred to as internal energy relaxation eigenvalues (IEREs). Those for slow decay correspond to the time scales for chemical reaction and are often referred to as chemically significant eigenvalues (CSEs). In general, for a system with n_1 wells and n_2 sources, the number of CSEs is equal to $n_1 + n_2$. If the system is conservative (i.e., it has no sinks/irreversible loss channels), then one of the CSEs will be zero within numerical precision.

10.1.1.5 Species Profiles

```
Print time dependent species and product profiles
{
```

Timestep/s	acetyl	Int1	Int2	lactone
1e-011	0.999999	1.3743e-006	1.59562e-010	2.89049e-011
1.25893e-011	0.999998	1.72938e-006	2.51184e-010	3.21681e-011
1.58489e-011	0.999998	2.17595e-006	3.95081e-010	3.7701e-011
1.99526e-011	0.999997	2.73745e-006	6.20583e-010	4.90837e-011
2.51189e-011	0.999997	3.44324e-006	9.73003e-010	7.0918e-011
3.16228e-011	0.999996	4.33003e-006	1.52194e-009	1.13004e-010
3.98107e-011	0.999995	5.44372e-006	2.37345e-009	1.94525e-010
5.01187e-011	0.999993	6.84152e-006	3.68781e-009	3.51826e-010
6.30957e-011	0.999991	8.59463e-006	5.70456e-009	6.51742e-010

This section gives time dependent species and product profiles of the system. The first column is the time step in seconds, and all the following columns are the populations at the specific time step of that species. For intermediates, these are calculated by simply summing the time dependent solutions for each energy grain which correspond to a particular isomer. The normalized time dependent product profiles are obtained in a related, but slightly more indirect manner. The normalized total product yield, $P(t)$, is as follows:

$$P(t) = 1 - \sum_{i}\sum_{E} \boldsymbol{p_i}(E,t)$$

Equation 4.5

where $\boldsymbol{p_i}(E,t)$ is the time dependent population of the energy grains spanning the entire state space of the *i*th well obtained from (4.4). If only one product channel is available, then (4.5) provides information regarding the time dependent product yield; however, when there are *n* products, then the normalized yield of a specific product at time *t*, $P_{ni}(t)$, may be written as:

$$P_{ni}(t) = \int_{\mathbf{0}}^{\boldsymbol{t}} \sum_{E\in i} k_{ni}(E)\boldsymbol{p_i}(E,t) = \sum_{E\in i} k_{ni}(E)[\boldsymbol{U\Lambda^{-1}}(\mathbf{1} - e^{\boldsymbol{\Lambda t}})\boldsymbol{U^{-1}p}(0)]$$

Equation 4.6

where $k_{pi}(E)$ are the microcanonical rate constants for formation of product *n* from isomer *i*.

10.1.1.6 Phenomenological rate coefficients

Bartis Widom eigenvalue/eigenvector analysis

This section gives phenomenological rate coefficients (i.e, *k*(*T*,*P*)) derived from an eigenvalue-eigenvector analysis of the solution to (E1). The mathematical development of the Bartis -Widom technique implemented is MESMER is described by Robertson *et al.*,[2] and so will not be detailed here. Briefly, the basic idea is as follows: the phenomenological rates for an arbitrary interconnected kinetic system may be described identically to (E2), with the primary difference being the absence of a description for collisional relaxation kinetics – i.e., all the rate coefficients correspond to interconversion between species. This is the sort of approach generally used to interpret kinetics experiments: for a system of *n* species, the kinetics of the system may be described using an *n* × *n* rate coefficient matrix ***K*** representing *n* coupled first order differential equations, where the matrix element K_{ab} is the rate coefficient $k_{b\rightarrow a}(T,P)$. Diagonalization of this rate matrix yields a solution to the coupled differential equations in terms of *n* eigenvalues and *n* eigenvectors. The Bartis-Widom method exploits the separation between the IEREs and CSEs: assuming that the CSEs obtained from the diagonalization of ***M*** (i.e., the full energy grained master equation, which includes a model for collisional relaxation) are identical those which would be obtained from diagonalization of ***K***, then the phenomenological rate matrix ***K*** may be obtained using simple matrix algebra. The Bartis-Widom analysis is a very powerful technique because it provides

a global description of the time dependent kinetics in terms of $n \times n$ rate coefficients, and in many cases, the phenomenological rate coefficient is the quantity of interest to be obtained from a ME calculation. However, the Bartis-Widom analysis relies on the separation between CSEs and IEREs. If these are not well separated by more than an order of magnitude, then MESMER will print a warning, and the user should proceed with caution because the Bartis Widom rate coefficients may not be reliable. In such cases, and as long as numerical precision is not an issue, the user may rely on the species profiles, since these do not require separation between CSEs and IEREs. When there is good separation (i.e., at least an order of magnitude) between the CSEs and the IEREs, then the species profiles printouts are identical to the species profiles that would be obtained from the phenomenological rate coefficients. MESMER prints out the following sections for any system, all of which are defined in Robertson *et al.*:

```
Z_matrix * Z_matrix^(-1):
```

The **Z** matrix is identical to that described in eq (19) of Robertson *et al.* The matrix printed below this header is essentially a test of numerical accuracy, and should give the identity matrix within numerical precision.

```
Kr matrix:
```

The $\boldsymbol{K_r}$ matrix is identical to that in Eq. (20) of Robertson *et al.*

```
First order & pseudo first order rate coefficients for isomerization rxns:
```

The pseudo and pseudo first order rate coefficients printed in this section are contained in the $\boldsymbol{K_r}$ matrix with labels describing the interconversion to which they correspond.

```
Kp matrix:
```

The $\boldsymbol{K_p}$ matrix is identical to that in eq (35) of Robertson *et al.*, and is only printed out for systems which have sinks / irreversible loss channels.

```
First order & pseudo first order rate coefficients for loss rxns:
```

The first order and pseudo first order rate coefficients printed in this section are contained in the $\boldsymbol{K_p}$ matrix with labels describing the interconversion to which they correspond.

10.1.2 mesmer.log

This file contains information, warning and error messages generated during the calculation. Also, it records when default values of input parameters have been used and

provides additional usage log of some variables when MESMER finishes calculation. It will be useful for anyone who is writing an input file from the beginning, to check this file for any missed arguments.

10.1.3 XML output

The XML output contains a copy of the input XML file together with:

- explicit entries for parameters where a default value was used;
- molecules, etc. from librarymols.xml and secondary input files that were used;
- calculated partition functions
- calculated microcanonical rate coefficients
- calculated Bartis-Widom rate coefficients. *When viewed in Firefox, a simplified version of the data suitable for cutting and pasting into speadsheets, etc. is available. This can also be produced by applying the file punch.xsl tothe output XML file, see section 4.2.4.*
- calculated species/ time profiles. *These are also presented graphically.*
- metadata, including name of user etc.

The calculated values are a subset of those in mesmer.test. The last two groups of data are in an additional element `<analysis>` under `<mesmer>` .

XML data can be re-formatted for various purposes, and a more human-friendly text presentation and an energy level diagram has been provided for the MESMER XML files using XSLT. Currently, it is necessary to use Firefox 3. See section 8.1.

10.2 Other files

10.2.1 defaults.xml

Many of the parameters specified in the input file come with default values. These may be properties of molecules or reactions, methods and modelling parameters. If MESMER requires a value and it is not present in the input file, it reads a value from the file `defaults.xml`, inserts the entry into the output XML file and records these actions in mesmer.log. If the default value could be contentious, it flags the need for user checking by

making the entries in mesmer.log and the output XML file in uppercase letters. For instance, spinMultiplicity has a default value of 1, which is not flagged in uppercase, while the default value of $\langle \Delta E \rangle_d$ of 130 cm^{-1} is. The `defaults.xml` file can be edited if necessary.

It is good practice to specify all the reactants and products of a reaction in a `<molecule>` element with a `ref` attribute. If there is no molecule under `<moleculeList>` with an `id` matching the `ref`, or if the matching molecule has no content, or if it fails to initialize, MESMER searches in the file `librarymols.xml` . If successful, it copies the molecule into the output XML file.

10.2.2 librarymols.xml

The intention is that `librarymols.xml` will contain many of the smaller common molecules, such as H, H_2O, NO, etc., to reduce the amount of data a user has to provide. This file is a in CML format and can be edited. There is a tutorial "Adding a molecule to the library.html" to illustrate how to extract a molecule with basic data (chemical structure, vibrational frequencies, energy) from a collection on a NIST website.

Molecules with an attribute `active=false` in this file are ignored, giving the opportunity to easily select alternatives.

Alternative names for molecules can be specified in `librarymols.xml`. for example:

```
<molecule id="oh" ref="OH">
```

Then, if a datafile referenced a molecule "oh", but had no complete specification of it, MESMER would insert the data from the molecule OH in `librarymols.xml`, but with an id="oh", so that this name can continue to be used in the datafile. It would have an additional attribute libId to show its origin.

These procedures mean that the output XML file is explicit: if used as input for further runs, no default values are used. The safest way of sharing data files between users, who might have adjusted the defaults, is to use an output XML file.

10.2.3 Secondary input files

Another facility for making use of a library of molecules is to use *secondary input files*. These are XML files specified on the command line after the main XML file. For instance, in

```
mesmer maininput.xml sec1.xml sec2.xml –o outfile.xml
```

sec1.xml and sec2.xml are secondary input files. MESMER inserts their contents into the main XML. So if the secondary file contained:

```
<moleculeList>
   <molecule id="mol1"> … </molecule>
   <molecule id="mol2"> … </molecule>
</moleculeList>
```

the molecules would be inserted into the main <moleculeList>. Or a secondary file like:

```
<me:control>
  <me:printSpeciesProfile />
</me:control>
```

might be used to obtain a different output, without editing the main data file.

10.2.4 source.dot and source.ps

These two files show the file dependency tree structure of the MESMER source code which is a guide for programmers to avoid repeated inclusion. File source.dot is produced by a PERL script cinclude2dot.pl (http://flourish.org/cinclude2dot/).

10.2.5 mesmer1.xsl, mesmerDiag.xsl, popDiag.xsl and switchcontent.xsl

These files, located in the root folder, provide a browser interface to XML files, where xsl stands for extensible stylesheet language. The MESMER *.xml input file refers to the *.xsl stylesheets using a relative pathname. In order to view the *.xml file in Firefox, one must verify that the XML input files refer to the correct location of the *.xsl files. The default for the input files included in the QA directory is two levels down in the root folder of MESMER; however, if a *.xml input file is in a different location, one can change the href path in second line of the XML file

```
<?xml-stylesheet type='text/xsl' href='../../mesmer1.xsl'?>
```

so that it points to the correct location of the *.xsl files.

10.2.6 punch.xsl, punchout.bat

To make transfer of data to a spreadsheet (or other program) easier, the Bartis-Widom output data can be converted to a simplified comma-separated csv form. When viewed in Firefox the Bartis-Widom has a facility to display the data in this form so that it can be copied and pasted.

Alternatively, the XSLT file punch.xsl can be applied to the output XML file in an external program, such as saxon or expat. For Windows systems, a batch file punchout.bat will run

msxsl.exe, which can be downloaded from the Microsoft website. Adding a shortcut to punchout.bat to the SendTo folder allows it to be run from Windows Explorer without having to open a command window.

11 Test Suite

As discussed previously, the `MesmerQA` folder contains a number of test jobs. These jobs are short in duration and are designed to assist in the development process of MESMER in that they can be run at regular intervals during development as a regression check. These jobs can be used as templates for constructing other jobs and, while there is no one job that includes all the MESMER functionality discussed thus far, taken together, these jobs do utilize most of MESMER's functionality. So users should be able to patch together their own MESMER input files using fragments of these input files.

Also supplied with MESMER is a folder called `examples` which is contains a number of examples that are currently being developed. These jobs tend to be of longer duration than those in `MesmerQA`, e.g. among them there are jobs that exercise the fitting facility within MESMER. These jobs also provide a resource that users might use to construct their own jobs.

11.1 MesmerQA

Below, we briefly describe each of the test jobs, and include a screen shot of the corresponding PES, visualized using Firefox to read the *.xml input file. Note that some of the QA directories include additional input files beyond those discussed in this part of the manual. These files are in various stages of development by the MESMER authors. While they are not discussed explicitly below, there's no harm in experimenting with them!

11.1.1 N-Pentyl Isomerization

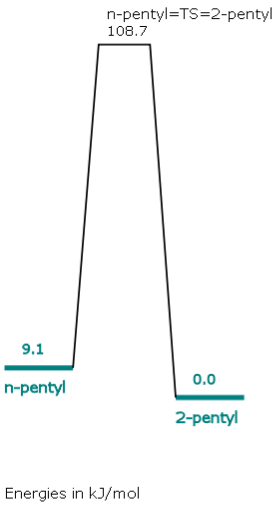

Figure 6.2: PES schematic for N-Pentyl Isomerization

The input file for this job is pentyl_isomerization_test.xml. This is a two well system for the isomerization of 2-pentyl over n-pentyl in an Argon bath gas using a standard RRKM treatment at the transition state.

11.1.2 Cyclopropene Isomerization + Reservoir State

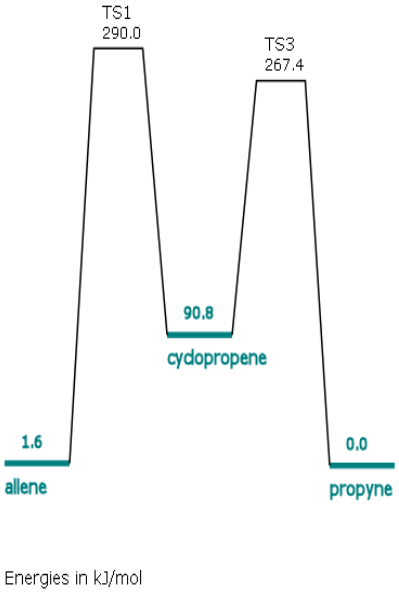

Figure 6.3: schematic of the PES used the cyclopropene isomerization test

The Cyclopropene_isomerization_test.xml input file is located within the folder ‘cyclopropene isomerization/’. This is a three well system for the isomerization of allene, cyclopropene, and propyne in a He bath gas using a standard RRKM treatment at both transition states.

The Cyclopropene_isomerization_reservoir_state_test.xml input file is located within the folder ‘cyclopropene isomerization reservoir state/’. This is a three well system for the isomerization of allene, cyclopropene, and propyne in a He bath gas using a standard RRKM treatment at both transition states; however, it includes reservoir states for each isomer.

11.1.3 H + SO2

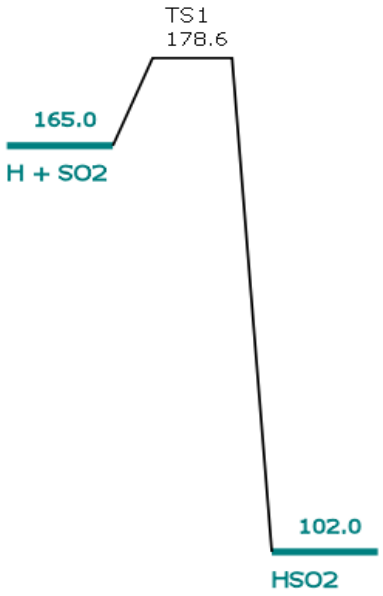

Figure 6.4: PES schematic for H + SO_2 test system

The HSO2_test.xml input file is located within the folder ‘HSO2/’. This system includes one well and a bimolecular source term, which is composed of one deficient and one excess reactant in an Ar bath gas using a standard RRKM treatment at TS1.

11.1.4 OH + C2H2

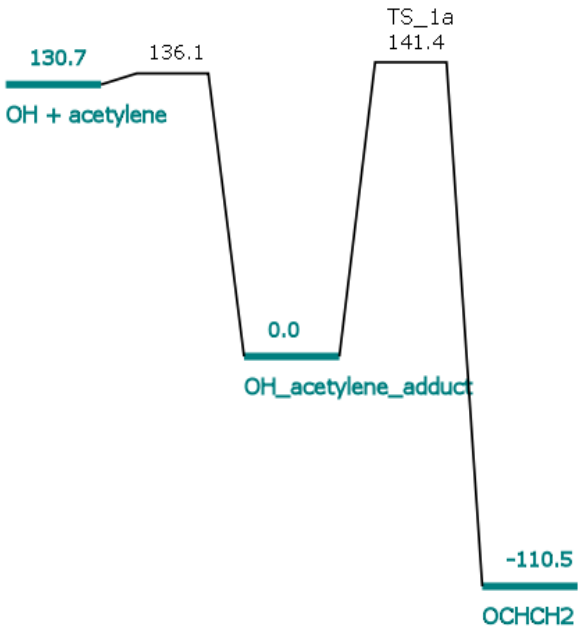

Figure 6.5: PES for OH + C_2H_2 test job

The OH_acetylene_association_test.xml input file is located within the folder 'OH acetylene association/'. This system includes one well, a bimolecular source term, and one irreversible unimolecular channel via TS_1a with an $OCHCH_2$ sink in an N_2 bath gas. TS_1a is treated using standard RRKM theory, and the association TS is treated using a standard ILT.

11.1.5 O2 + CH3CO

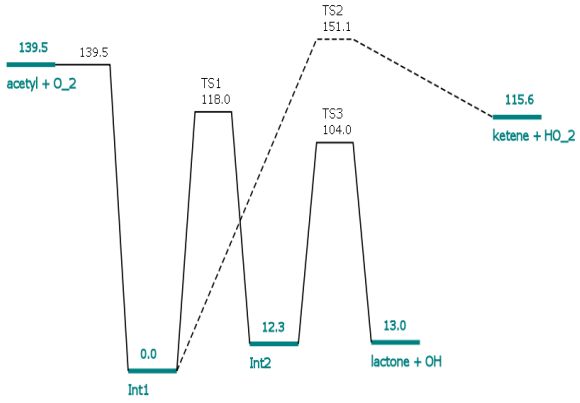

Figure 6.6: PES for O_2 + CH_3CO

The Acetyl_O2_association.xml input file is located within the folder 'Acetyl O2 association/'. This system includes two wells, a bimolecular source term, and two irreversible unimolecular dissociation channels via both TS2 and TS3, and He bath gas. TS1 is treated using standard RRKM theory with an Eckart tunnelling correction, the association TS is treated using a standard ILT, and all other TSs with standard RRKM theory.

11.1.6 i-propyl

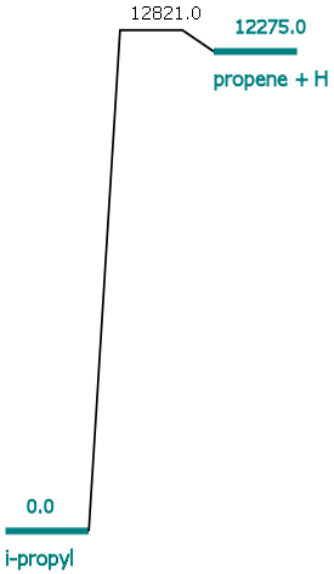

Figure 6.7: PES for i-propyl dissociation

The ipropyl_test.xml input file is located within the folder ‘i-propyl/’. This system includes one well, and an irreversible unimolecular dissociation channel in a He bath gas. The $k(E)$s for the irreversible dissociation channel are calculated using a reverse ILT of the propene + H association rate coefficients.

11.1.7 Thermodynamic Table

The ThermodynamicTable.xml input file in the folder ThermodynamicTable generates thermodynamics function date for the species involved in the reaction $OH + NO \rightarrow HONO$. In this the input file the OH radical is defined using the `DefinedStatesRotors` class.

11.2 Examples

Some of the systems in the examples folder are discussed below. The list is not complete as it is expected that the number systems will increase with time.

11.2.1 Benzene-OH Oxidation

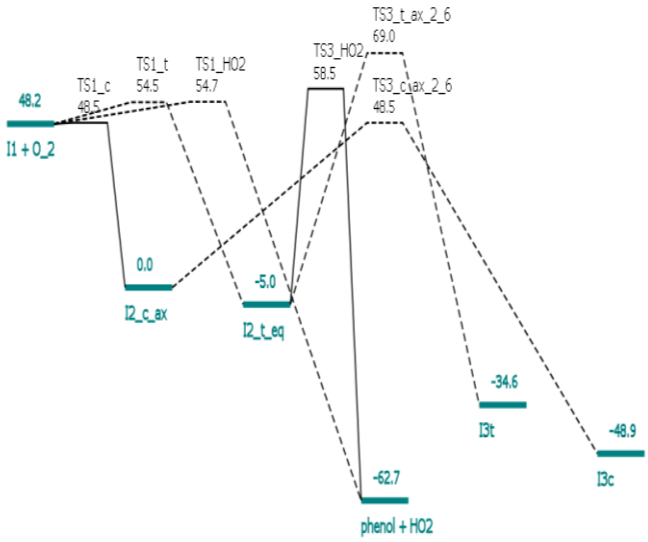

Figure 6.8: PES for benzene-OH + O_2

The benzene_oxidation_test.xml input file is located within the folder ‘benzene_oxidation/’. This system includes a bimolecular source (I1 + O2), two wells (I2_c_ax and I2_t_eq), and four irreversible channels in N_2 bath gas. Two of the irreversible channels are unimolecular isomerizations (to I3t and I3c), one is a unimolecular dissociation

(I2_t_eq → phenol + HO_2), and one is an irreversible exchange reaction (benzene-OH + O_2 → phenol + HO_2). The *k*(*E*)s for each channel are calculated using RRKM theory.

11.2.2 i-propyl

This example extends that the MesmerQA job of the same name so as to perform a fitting of the experimental data obtained by Seakins et al (*J. Phys. Chem.*, **97**, 4450 (1993)).

11.2.3 Spin Forbidden Test Systems

Input files that show how to invoke spin forbidden RRKM theory are include in the folder 'spin_forbidden_kinetics/'. This folder includes two different input files, one of which invokes Landau Zener corrections (LZ_test.xml), and the other of which invokes WKB corrections (WKB_test.xml), as described in Harvey and Aschi, *Faraday Discuss*., 2003(124), 129-143. The modelled systems are simple isomerizations of singlet cyclopentyne to the lowest lying triplet via the DFT calculated minimum energy crossing point (MECP).

12 Adding Functionality to MESMER

MESMER has been written in a manner than anticipates future developments, so that as the functionality of MESMER increases, it is expected that this portion of the manual to grow. For example, we can envision the implementation of more sophisticated models for the calculation of tunnelling corrections, spin forbidden transmission coefficients, collisional energy transfer, density of states calculations, non-equilibrium initial distributions, and microcanonical rate coefficients.

Many of the features within MESMER has been written with the use of 'plug-in' classes – this exploits the polymorphic feature of object-oriented languages such as C++. Plug-in classes are concrete classes derived from abstract base classes, and may be added without changing any of the existing code. The abstract base classes expose an interface to the rest of MESMER and derived class must conform to this interface in order to work with MESMER. The constructor of a plug-in class registers a class instance (object) with the base class. A pointer to a derived plug-in object may be obtained by supplying the id (i.e., a string) of the derived class to the Find function on the base class. To assist in the development of plug-in classes, the code for the energy transfer model describing and exponential down model has been extensively commented as a guide. The code in located in the files:

- `...\src\EnergyTransferModel.h`: this file contains the definition of the abstract base class for energy transfer, `EnergyTransferModel`.
- `...\src\plugins\ExponentialDown.cpp`: contains the definition and the implementation of the `ExponentialDown` class, which calculates energy transfer probabilties based on the exponential down model.

12.1 Data Access

It will likely be the case that new plug-in classes will require the user to supply new data in the input file, as a consequence it is a design feature of MESMER that plug-in classes should read their own data. This will allow additions to the MESMER code in a modular and relatively painless fashion. Additionally, it will minimize the amount of code that needs to be changed and retain backward compatibility with existing data structures and input files.

Input data is read into and stored in internal structures that reflect the data hierarchy set out in the input file. Access to and navigation through this data structure is via set of methods exposed on the pointer class of type `PersistPtr`. The set of methods are collected on an interface class `IPersist` and documented in the header file `Persistence.h`. When reading data some the methods that may be useful are:

12.1.1 XmlMoveTo

```
PersistPtr XmlMoveTo(const std::string& name)const
```

Returns a PersistPtr which can be used to read further down the input or output data. This method will try to locate the first child element with target name `name` and if this fails it will look for the first sibling with this target name. If this fails it will return `NULL`.

`name`: The name of the target element.

12.1.2 XmlRead

```
char* XmlRead()const
```

Returns the next item from the input document, that is the value of the current element or `NULL` if it has no value.

12.1.3 XmlReadValue

```
const char* XmlReadValue(const std::string& name, bool MustBeThere=true)
```

This method first looks to see if there is a child element of the specified name and if there is returns its value. If this fails it then looks to see if there is a corresponding attribute and returns this value. If `MustBeThere` is true(the default) then this method will try to insert a default value from defaults.xml and return, otherwise it returns `NULL`. If `name` is empty, returns `NULL` if there are no children and an empty string if there are.

`name`: The name of the target element or attribute.

`MustBeThere`: Optional argument indicating if the parameter is mandatory. The enum `optional` is a synonym for `true` and its use makes the code more understandable.

12.1.4 XmlReadDouble

```
double XmlReadDouble(const std::string& name, bool MustBeThere=true)
```

This method is the same as `XmlReadValue` but attempts to convert the value to type `double`. If there is no child element or attribute with the specified name then NaN (not a number) is returned.

`name`: The name of the target element.

`MustBeThere`: Optional argument indicating if the parameter is mandatory.

12.1.5 XmlReadInteger

```
int XmlReadInteger(const std::string& name, bool MustBeThere=true)
```

This method is the same as `XmlReadValue` but attempts to convert the value to type `int`.

`name`: The name of the target element.

`MustBeThere`: Optional argument indicating if the parameter is mandatory.

12.1.6 XmlReadBoolean

```
bool XmlReadBoolean( const std::string& name)
```

This method is the same as `XmlReadValue` but returns true if datatext associated with name is "1", "true", “yes” or nothing and returns false if datatext is something else or if element is not found.

`name`: The name of the target element.

There are also methods, similar to the above, for reading data from CML property elements.

12.2 Plug-in Classes

12.2.1 Calculation Methods

The abstract base class for calculation methods is `CalcMethod`, and is defined in `calcmethod.h`. The derived concrete classes offer different methods for overall calculation execution as follows:

`simpleCalc`: the default, and does a normal set of ME calculations at each of the specified pressure and temperature points.

`gridSearch`: initiates a grid search over parameter values specified by the user. χ^2 is calculated, against user specified experimental data, for every point in parameter space

specified by the user. Typically, data generated is plotted as a contour map to help located the minimum on the χ^2 surface.

`fitting`: This class determines the best fit parameters by optimizing the χ^2 surface and is based on Powell's methods in conjunction with golden section line searches to find the minimum in the χ^2 surface. The parameters that are to be optimized are specified in the same way as for the `gridSearch` method, the `upper` and `lower` limits specify the region of parameter space that will be explored by the search, the `stepsize` is ignored. The nature of the algorithm is iterative and the number of iterations are set via the parameter `<me:fittingIterations>2</me:fittingIterations>`.

`marquardt`: This class determines the best fit parameters by optimizing the χ^2 surface and is based on the Levenberg-Marquardt algorithm. Because rate data are derived from eigenvalues, analytic derivatives of the rate coefficients with respect to the parameters are not available and so numerical derivatives are used instead. The accuracy of the numerical derivatives is controlled by the input parameter `<me:MarquardtDerivDelta>1.e-03</me:MarquardtDerivDelta>`, which is used in a simple two point estimates of the numerical derivatives. The parameters that are to be optimized are specified in the same way as for the `gridSearch` method, the `upper` and `lower` limits specify the region of parameter space that will be explored by the search, the `stepsize` is ignored. As with the `fitting` calculation, the `marquardt` calculation is iterative and the number of iterations are set via `<me:MarquardtIterations>20</me:MarquardtIterations>`. In addition to limiting the number iterations, the calculation can also be controlled by the specification of a tolerance, `<me:MarquardtTolerance>0.01</me:MarquardtTolerance>` which will terminate the calculation when the relative change in the χ^2 value is less than the specified value.

An obvious question is which fitting method should be used? Experience suggests that the `marquardt` class tends to find the χ^2 minimum more rapidly and as a consequence can handle more parameters in a shorter time. The `fitting` class is slow but in some situations is the more robust and should be considered if the `marquardt` fails. Both classes calculate an estimate of the covariance matrix, from which parameter errors are extracted, and calculate the χ^2 statistic in order to give a goodness of fit measure. This data is written to the log file.

`ThermodynamicTable`: This class calculates the thermodynamic functions enthalpy (H(T)), entropy (S(T)) and Gibbs free energy (G(T)) of all the molecules defined in the

molecule list of an input file at user defined temperatures. The number of temperatures, the temperature interval and the units in which the thermodynamic functions are written can be specified as attributes, e.g.

```
<me:calcMethod me:NumberOfTemp="20" me:TempInterval="100.0" me:Units="kcal/Mol" >
ThermodynamicTable</me:calcMethod>
```

At present the allowed units are kcal/Mol and kJ/Mol. Values for the thermodynamic functions at the temperature of 298.15 K are always written. The tables are written to the .test output file.

12.2.2 Collisional Energy Transfer Models

The abstract base class for calculating energy transfer probabilities between grains is `EnergyTransferModel`, which lives in `EnergyTransferModel.h`. The derived concrete classes (i.e., the plug-in classes) offer different methods for calculating energy transfer probabilities. At present only the exponential down model is implemented:

`Exponential Down`: This calculates energy transfer probabilities on the basis of the exponential down model, the probability of transition from a grain of energy E to one of lower energy E' is given by,

$$P(E|E') = A(E')\exp(-(E' - E)/\langle\Delta E\rangle_d)$$

where $A(E')$ is a normalization factor. The probabilities of activating collisions are found by detailed balance.

It is often the case that $\langle\Delta E\rangle_d$ depends on temperature. In the MESMER implementation of the exponential down model the temperature dependence of $\langle\Delta E\rangle_d$ is modelled as:

$$\langle\Delta E\rangle_d = \langle\Delta E\rangle_{d,ref}\left(\frac{T}{T_{ref}}\right)^n$$

where T is temperature, T_{ref} is the reference temperature, and n is an exponent governing the temperature dependence. By default, n = 0, which means there is no temperature dependency for $\langle\Delta E\rangle_d$, so that $\langle\Delta E\rangle_d = \langle\Delta E\rangle_{d,ref}$. The user can change these values by the following syntax

```
<property dictRef="me:deltaEDown">
  <scalar units="cm-1" referenceTemperature="298" exponent="1.0">150.0</scalar>
```

```
</property>
```

This gives linear dependency of $\langle \Delta E \rangle_d$ in temperature with T_{ref} = 298, n = 1, and $\langle \Delta E \rangle_{d,ref}$ = 150 cm^{-1}. The parameters $\langle \Delta E \rangle_{d,ref}$ and n can both be floated as part of a fitting exercise: to float $\langle \Delta E \rangle_{d,ref}$ use the syntax,

```
<property dictRef="me:deltaEDown">
  <scalar units="cm-1" lower="140.0" upper="200.1" stepsize="10.0" >150.0</scalar>
</property>
```

and to float n use,

```
<property dictRef="me:deltaEDownTExponent">
  <scalar lower="0.0" upper="1.0" stepsize="0.01" >0.2</scalar>
</property>
```

12.2.3 Density of States

`DensityOfStatesCalculator` is the abstract base class for performing density of states calculations, and lives in `DensityOfStates.h`. The derived concrete classes (i.e., the plug-in classes) offer different methods for calculating densities of states for specific degrees of freedom that are convolved with densities of states derived from the other degrees of freedom to give a total density of states. These classes include:

`HinderedRotorQM1D`: this class calculates the quantum mechanical energy levels of a hindered internal rotor. This is done by expressing the hamiltonian for a hindered one-dimensional rotor in a basis set of one-dimensional rotational functions and diagonalizing. The effective mass to be used is the reduced moment of inertia about the bond that defines the internal rotation. This requires the bond to be idenitified and this is done with the input term `<bondRef>b1</bondRef>` were `b1`is the ID of a bond in the bond section defining the molecule. It is also necessary to define a hindering potential which can be done in one of two ways: as an `analytical` potential, defined as a fourier cosine expansion e.g. the simple potential (which might be used to model the rotation of a methyl group.

$$V(\theta) = \frac{V_0}{2}(1 - \cos 3\theta)$$

where the barrier height is 12.4 kJ/mol can be represented as:

```
<me:HinderedRotorPotential format="analytical" units="kJ/mol">
  <me:PotentialPoint index="0"  coefficient="6.2"/>
  <me:PotentialPoint index="3"  coefficient="-6.2"/>
```

```
        </me:HinderedRotorPotential>
```

where `index` repesents the value of the non-zero coefficints in the cosine expansion and `coefficient` the associated cosine expansion cofficient.

Alternatively, the hindering potential can be expressed in `numerical` format, as a set of potential points at regular intervals of the rotational coordinate, which may have been obtained from an *ab initio* calculation. These points are used to obtain a cosine expansion which is then used to determine the energy levels. The length of the cosine expansion is determined by the parameter `expansionSize` attribute. The units in which the potential are defined can be specified using the `units` attribute, at present the allowed units are `cm-1`, `kJ/mol`, `kcal/mol` and `Hartree`. The cosine expansion is usually sufficient, but occasionally a hindering potential might have a significant asymmetric character. In this situation a cosine expansion will not be sufficient and addition terms bases on sine functions are required and this can be added by specifying the attribute `UseSineTerms="yes"`, which will add to the expansion the same number of sine terms as specified by the `expansionSize` attribute. Note that using sine terms in the potential expansion has the consequence that the Hamiltonian is now represented as a complex matrix and this increases the time and space need to diagonalize it. A full definition of a numerical potential might look something like this:

```
      <me:ExtraDOSCMethod name="HinderedRotorQM1D">
        <bondRef>b8</bondRef>
        <me:HinderedRotorPotential format="numerical" units="Hartree" expansionSize="7"
UseSineTerms="yes">
          <me:PotentialPoint angle=   "0"      potential=   "-304.7521737" />
          <me:PotentialPoint angle=   "10"     potential=   "-304.7522079" />
          <me:PotentialPoint angle=   "20"     potential=   "-304.7524228" />
           ...
          <me:PotentialPoint angle=   "330"    potential=   "-304.7530499" />
          <me:PotentialPoint angle=   "340"    potential=   "-304.7526519" />
          <me:PotentialPoint angle=   "350"    potential=   "-304.7523336" />

        </me:HinderedRotorPotential>
      </me:ExtraDOSCMethod>
```

Some internal rotations pose symmetry e.g. the rotation of a methyl group. The exchange of similar particles imposes restrictions on the acceptable wave functions and as a consequence the number of states allowed is less than those calculated in MESMER, this can be approximately rectified by applying a correction factor, i.e. dividing by the periodicity of the hindered rotor potential and this can be specified with `<me:periodicity>3</me:periodicity>`. Note this symmetry factor is sometimes accounted for in the symmetry number of overall rotation and care should be taken that it is not accounted for twice.

DefinedStatesRotors: There are occassionally situations where the density of states cannot be easily factored into a series of convolutions, typically because of coupling terms between the generalized coordinates that are being used to define the Hamiltonian that describes the system being investigated. For this reason MESMER provides a class that allows specific state manifolds to be defined and which can then be convolved with other decoupled modes to give the overall density states. An example of the use of this class is the coupling between the electronic and rotational degrees of freedom of the OH radical which cannot be de-couple. The "ThermodynamicTable" example in the MesmerQA area (see below) shows how this class can be used.

12.2.4 Microcanonical Rates

The abstract base class for calculating microcanoncial rates is `MicroRateCalculator`, which lives in `Microrate.h`. The derived concrete classes (i.e., the plug-in classes) offer different methods for calculating microcanonical rate coefficients as follows:

MesmerILT: which includes standard ILT, unimolecular ILT, or reverse ILT

SimpleRRKM: which uses the standard RRKM equation

SimpleILT: which is not presently used for anything, and was initially used for testing purposes

12.2.5 Tunneling Corrections

The abstract base class for tunnelling corrections is `TunnelingCalculator`, which lives in `Tunneling.h`. The derived concrete classes (i.e., the plug-in classes) offer different methods for calculating tunnelling corrections as follows:

EckartCoefficients: the is class approximates tunnelling using a one dimensional asymmetric Eckart potential using the method described by Miller,[1] and specified in MESMER as `<me:tunneling>Eckart</me:tunneling>` in the section(s) defining the reacion(s) to which tunneling is to be applied. A key parameter in this method is the imaginary frequency associated with the reaction coordinate. This parameter is specified in the definition of the transition state in the molecular list section of the input using the keyword `me:imFreqs`. MESMER allows this frequency to be floated in a fitting exercise. A typical definition might look something like:

```
<property dictRef="me:imFreqs">
```

```
    <scalar units="cm-1" lower="500" upper="600" stepsize="10">533.0</scalar>
</property>
```

WKBTunnelling: this class approximates tunnelling using a user defined potential according to a semiclassical WKB method. In order to implement this tunnelling correction the user is required to define a potential corresponding to the minimum energy path over the relevent transition state. This potential is then numericaly integrated to give the barrier penertration integral ϕ as described by Garrett and Truhlar, *J. Chem. Phys.*, 1979(83), 2921-2925. This potential is defined is defined after the <me:DOSCMethod> element for the molecule and has the form below:

```
<me:IRCPotential units="kcal/mol" ReducedMass="0.679">
   <me:PotentialPoint ReacCoord/m= "-3.5E-10" potential= "1.47"/>
   <me:PotentialPoint ReacCoord/m= "-3.4E-10" potential= "1.51"/>
   <me:PotentialPoint ReacCoord/m= "-3.3E-10" potential= "1.58"/>
```

This potential should be defined such that ReacCoord = 0 at threshold. The units of the reaction coordinate are meters and the units for the corresponding potential are defined by the units element. The reduced mass of the reaction coordinate may also be defined in units of amu though mass weighted coordinates are assumed if this element is not present.

12.2.6 Spin Forbidden RRKM theory

The abstract base class for tunnelling corrections is CrossingCalculator, which lives in Crossing.h. The derived concrete classes (i.e., the plug-in classes) offer different methods for calculating tunnelling corrections as follows:

LandauZenerCrossingCoeff: which calculates transmission coefficients using Landau Zener theory

WKBCrossingCoeff: which calculates transmission coefficients that include the effects of tunnelling below threshold

12.2.7 Distribution Calculator

The abstract base class for calculating the initial distributions within the grains is DistributionCalculator, which lives in Distribution.h. The derived concrete classes (i.e., the plug-in classes) offer different methods for the initial grain distribution as follows:

BoltzmannDistribution: which uses a Boltzmann population for the initial grain distribution within a particular well

In the future, we hope to add a plug-in class that will calculate initial non-equilibrium grain distributions, based on energy partitioning models similar to the prior distribution.

13 MESMER FAQs

Q. What is qtot? What is sumc? What is sumg?

This is discussed in section 10.1.1.1

Q. What is conc?

It is the number density in units of particles cm^{-3}

Q. For an irreversible Reaction –either an isomerization or a dissociation reaction – do I need to input the data for the product?

Irreversible reactions should not require any data apart from the name of the sink molecule, and the sink molecule requires no more than a molecule id within the moleculeList portion of the input. So if HO_2 is a sink molecule, and you can't supply its molecular data, then all you need to supply for HO_2 in the moleculeList is the following:

```
<molecule id="HO_2"/>
```

Q. What are the different types of reactions in MESMER and how are they defined?

MESMER has four types of reactions:

- AssociationReaction, which has two reactants (one excess and one deficient), and one product, which is a modelled molecule
- IrreversibleReaction, which has one modelled molecule reactant, and either one or two sink molecules for products
- IsomerizationReaction, which has one reactant and one product, each of which is a modelled molecule
- IrreversibleExchangeReaction, which has two reactants (one excess and one deficient), and two products, both of which are sink molecules

Q. Do all the energies in a MESMER input file need to be in the same units?

For ZPEs, yes, they do. The ZPE units for each stationary point must be consistent, and if you need energies for any other part of the calculation (e.g., an ILT), then the units for those must be consistent with the molecular ZPEs.

Q. What is the file naming convention of MESMER?

This is described in section 8.

Q. How can I tell if my calculation is suffering from numerical precision problems?

MESMER uses numerical algorithms to solve the Master Equation, and thus is not immune from the sorts of numerical issues that arise in any numerical approach. In general, for any system, there will be a temperature and pressure regime where the results output by MESMER are not reliable; however, the onset of this unreliability varies from system to system. Certain portions of the MESMER output are more resistant to numerical problems than others: for example, the product yield profiles vs. time tend to be the least reliable, since their calculation involves a great deal of arithmetic. The rate coefficients output in the Bartis-Widom analysis tend to be reliable over a relatively larger range of temperatures and pressures. The only way to systematically determine whether MESMER results are reliable is to run ME calculations over a range of temperatures and pressures, plot the results of interest, and examine where the MESMER output looks nonsensical. For example, in many applications, rate coefficients are the desired output, so the user should plot fall off curves at several temperatures, and they will see where the results numerically blow up. MESMER includes some facilities for circumventing numerical issues, and they are detailed in this manual.

Q. Do I have to specify an atom and a bond array?

Perhaps. MESMER requires neither an atom nor a bond array to carry out a ME calculation. However, if there is no molecular weight or rotational constants specified, they can be calculated from the chemical structure if it is provided. These data structures are routinely present if a structure is imported from a computational chemistry program via

OpenBabel, and may offer useful functionality in the future (e.g., visualizing molecular structures in Firefox).

14 Theoretical Background

This section is not meant to be a thorough mathematical description of the Master Equation, but rather to provide broad overview of MESMER, and provide insight into some of the less straightforward details in MESMER.

For thorough reviews of the ME mathematical development in MESMER, we refer the readers to work by Pilling and Robertson,[2-4] and Miller and Klippenstein.[5,6] Useful discussions of numerical precision issues may be found in Gannon et al.[7] and references therein, and some discussion of the Standard ILT is found in Davies *et al.*[8]

14.1 Matrix Formulation of the EGME

The form of the EGME in MESMER is the one dimensional ME, wherein the total rovibrational energy of the system, *E*, is the independent variable. Indeed, other forms of the EGME consider the time dependent evolution of the system with respect to the total *E* as well as angular momentum, *J*. However, such 2 dimensional ME treatments are restricted in their application, given the difficulty of describing the transition probabilities wherein both *E* and *J* are coupled.[6] They may only be used to solve the ME in the collisionless limit, or for a system that has a single isomer.[9,10] Thus, the bulk of ME modelling for systems under conditions of relevance to atmospheric and combustion chemistry is restricted to a one dimensional ME.[6]

In the mathematical formulation of the one dimensional EGME, the population of rovibrational energy levels in different isomers (denoted by subscript *i*) on the potential energy surface are lumped into energy grains, characterized by an average energy, E_i, and the population in each grain, $n_i(E_i, t)$, is described by a set of coupled differential equations that account for collisional energy transfer within each isomer as well as isomerization and dissociation:

$$\frac{d}{dt}n_i(E_i,t) = \omega \int_{E_{i,0}}^{\infty} P(E_i|E_i')\, n_i(E_i',t)dE_i' - \omega n_i(E_i,t) - \sum_{j\neq i}^{M} k_{ji}(E_i)\, n_i(E_i,t)$$
$$+ \sum_{j\neq i}^{M} k_{ij}(E_j)\, n_j(E_j,t) - k_{Pi}(E_i)n_i(E_i,t) - k_{Ri}(E_i)n_i(E_i,t)$$
$$+ k_{Ri}(E_i)K_{Ri}^{eq}\frac{\rho_i(E_i)e^{-\beta E_i}}{Q_i(\beta)}n_R n_m$$

Equation 9.1

The first term in Eq 9.1 represents the probability that $n_i(E_i,t)$ is populated by collisional energy transfer via activating/deactivating bath gas collisions. ω is the Leonard-Jones collision frequency,[11] and $P(E_i|E_i')$ is the probability that collision with bath gas will result in a transition from a grain with energy E_i' to a grain with energy E_i. The second term represents the loss from grain E_i via collisional energy transfer. The third term represents the loss from grain E_i via reaction to give other isomers, denoted by subscript *j*. $k_{ji}(E_i)$ is the microcanonical rate constant for loss from isomer *i* to isomer *j*. The fourth term represents the population of grain E_i by reactions from isomer *j* that give isomer *i* , the grains E_i and E_j spanning the same range energy range. The fifth term represents the rate of loss from grain E_i via dissociation to products, with $k_{Pi}(E_i)$, representing the corresponding rate of loss. Because re-association of the products of unimolecular dissociation is generally negligible on an experimental time scale, dissociation to products is often treated via an infinite sink approximation – i.e., re-association is not considered. The final two terms are associated with the so-called bimolecular source term. They only apply to those wells that are populated via bimolecular association. Assuming that the reactants are thermalized via bath gas collisions in a Boltzmann distribution, and that a pseudo-first order approximation is appropriate, then the sixth term and seventh term represent the rate at which two reactants associate to populate grain E_i, and the rate of loss from that grain via re-dissociation to reactants, respectively. $k_{Ri}(E_i)$ represents the rate constant at which E_i re-dissociates to give reactants, *R*, and K_{Ri}^{eq} is the equilibrium constant between isomer *i* and the reactants. $Q_i(T) = \sum \rho_i(E_i)e^{-\beta E}$, which is the rovibrational partition function for the molecular species corresponding to isomer *i*.

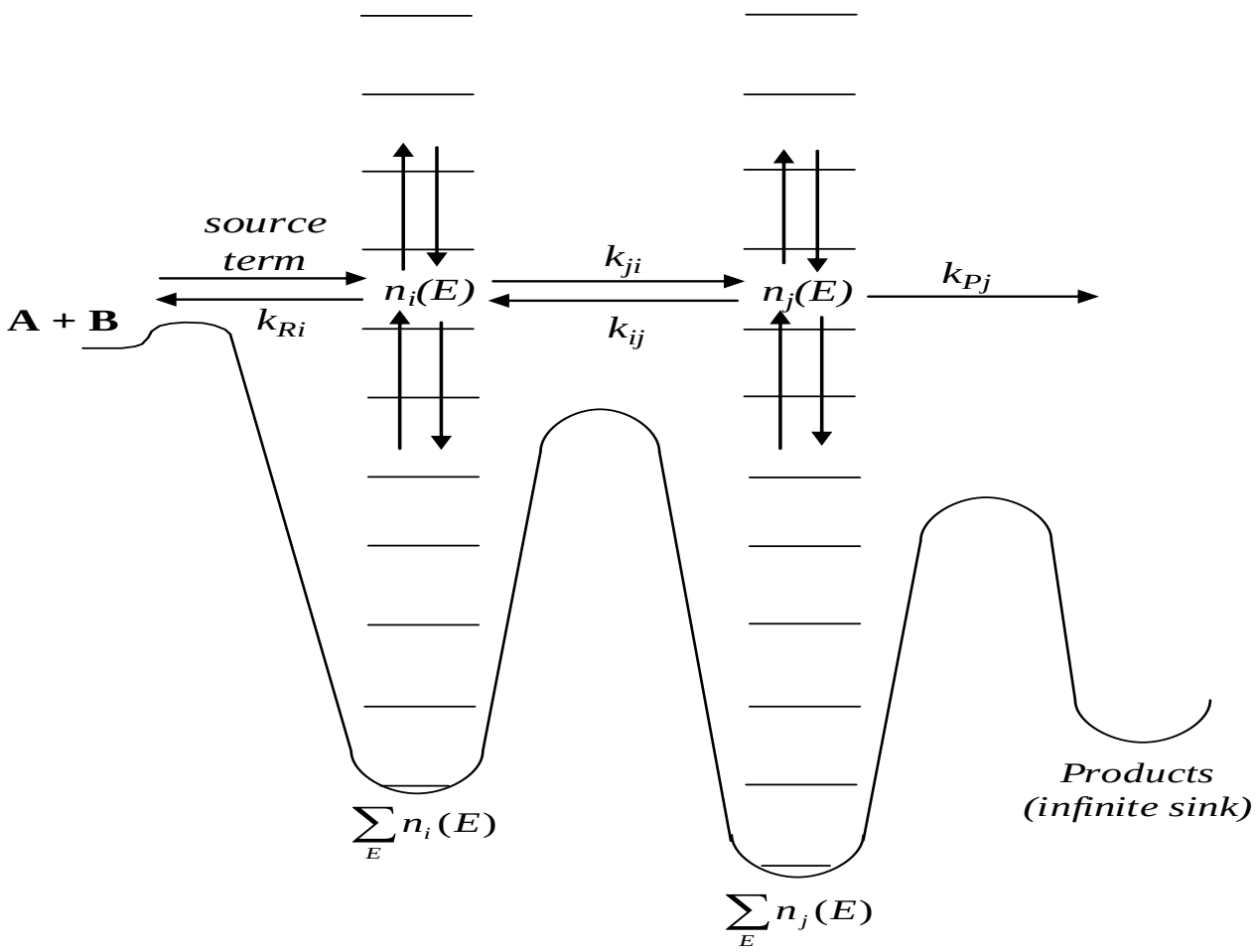

Figure 9.9: pictoral representation of Eq 1.9 for a two well system, composed of isomers *i* and *j*. The reactants, A + B, are connected to isomer *i*, and the product channel is connected to isomer *j*. This figure shows the terms of Eq 1.9 as they apply to one grain within each isomer, $n_i(E_i)$ and $n_j(E_j)$.

Microcanonical rate coefficients for the unimolecular reactions that occur in each energy grain, $k(E)$, are calculated from the PES data pertaining to the reagents and transition states via the microcanonical transition state/RRKM theory expression:

$$k(E) = \frac{W(E)}{h\rho(E)}$$

Equation 9.2

where $W(E)$ is the sum of rovibrational states at the optimized transition state geometry and $\rho(E)$ is the density of rovibrational states of the isomer. As discussed by Baer and Hase,[12] RRKM theory depends on the assumption that the total phase space of a molecule is populated statistically. That is, the population density of molecules in phase space may be assumed uniform as the molecule moves from the reactant phase space, through the transition state dividing surface, and into the product phase space. This is equivalent to assuming that

all regions of phase space are available to the molecule – i.e., that the time scale for energy randomization within a molecule is very fast with respect to the rate of reaction, maintaining a microcanonical ensemble. This common assumption, which is fundamental to ME analysis, is called the ergodicity assumption.

Eq. 9.2 is applicable for transition state dividing surfaces located at constrained geometry with a well defined energetic barrier. When the reaction in question is barrierless, a first principles determination of $k(E)$ requires a variational approach – i.e., $k(E)$ are calculated by minimizing $W(E)$ on the PES in question. An alternative approach, available in Mesmer, is to calculate the $k_{Ri}(E)$s for barrierless association reactions using an inverse Laplace transform (ILT).[8,13] With this technique, the microcanonical dissociation $k_{Ri}(E)$s are determined from experimental measurements of the temperature dependent high pressure limiting rate coefficient for association. The microcanonical rate coefficients for dissociation are then determined via detailed balance, as indicated in Eq. 9.1.

In section the plug-in class for exponential down transition probabilities was:

$$P(E|E') = A(E')\exp(-(E' - E)/\langle \Delta E \rangle_d)$$

Equation 9.3

where $E' > E$, $A(E')$ is a normalization constant, and $\langle \Delta E \rangle_d$ is the average energy transferred per collision in a downward direction. The transition probabilities for energy transfer in the upward direction may be obtained from those calculated with Eq. 9.3 via detailed balance. The exponential down model is the most commonly used for describing collisional energy transfer probabilities. It derives from scattering theory, and it reflects the common sense notion that collisions which transfer lots of energy are less probable than those that transfer small amounts of energy.[3] Other models with different transition probability distributions have been proposed,[3] such as Gaussian models[11] and double exponential models.[6] In a recent review of ME techniques by Miller and Klippenstein, they noted that classical trajectory simulations as well as experimental data suggest that the exponential down model is not the most accurate for describing collisional transition probabilities.[6] Models that produce longer tails are more accurate.

However, nearly all those ME models published in the literature utilize the single exponential down model. Other functions, such as a double exponential model, feature more parameters and systematic techniques for assigning parameter values have not been

established. Additionally, given the extensive use of the single exponential down model in the literature, a set of typical $\langle \Delta E \rangle_d$ values has emerged. For example, at room temperature He bath gas tends to have $\langle \Delta E \rangle_d$ values from ~120-175 cm^{-1}, while O_2 and N_2 bath gases tend to have $\langle \Delta E \rangle_d$ values of ~175-275 cm^{-1}. In general, $\langle \Delta E \rangle_d$ is left as a variable parameter determined by fitting to experimental data, within the limits given above, and it usually shows a slight positive temperature dependence in one dimensional ME analyses. The origin of the temperature dependence is not entirely clear, although Miller suggested that this temperature dependence may correspond to rotational excitation.[10] Classical Trajectory calculations have identified the dependence of $\langle \Delta E \rangle_d$ on the angular momentum of the target molecule. Experimentally, higher temperatures correspond to higher angular momentum states, and in the 1d ME, this is manifest as an effective increase in $\langle \Delta E \rangle_d$.[10]

The collisional energy transfer models discussed above assume that the transition probability depends only on the energy of the molecule, and not on its collisional history or its configuration. The same is true of the RRKM $k(E)$s. Such processes, where the probability that a system occupies a particular state, depends only on the immediately previous state and nothing else, are common to many fields, and are referred to as Markov processes. Indeed, the ME in Eq 9.1 is a stochastic differential equation of the Markov type, and its relationship with the more general field of stochastic probability theory has been discussed in detail by Pilling and Robertson.[3] So long as the time that a pair of molecules spends in collision is much smaller than the time between collisions, the molecular configuration in a collision event should not depend on the molecular configuration in the previous collision event. The energy transferred in a collision may be treated as depending only on the internal energy of the molecule, and not its configurational history – i.e., collisional energy transfer may be treated as a 'random walk', and the ME is applicable.[3,6]

The coupled stochastic differential equations represented by Eq 9.1 may be reformulated as:

$$\frac{d}{dt}\boldsymbol{n} = \boldsymbol{M}\boldsymbol{n}$$

Equation 9.4

where $\boldsymbol{n}$ is a vector containing the populations of the grains for each isomer, $n_i(E_i, t)$, and **M** is the matrix that describes collisional energy transfer, as well as reactive loss and gain for

each grain. In the case where Eq 8.1 includes a pseudo-first order bimolecular source term to describe the fractional rates of population of the entrance well by the reactants, then the final element of $\boldsymbol{n}$ corresponds to the time dependent population of the reactant that is not in excess. Eq. 8.4 shows that the solution of the ME has been reduced to a standard eigenvalue problem, and diagonalization of $\boldsymbol{M}$ yields the corresponding eigenpair solutions.

Approximating the time dependent grain populations as sums of exponential functions, and combining the eigenpair solutions with the appropriate initial conditions vector for describing the system at time zero, the time dependent grain evolution for a particular isomer, $n_i(E_i, t)$, may be obtained. The solution to the ME may be written as:

$$\boldsymbol{n}(t) = \boldsymbol{U}e^{\boldsymbol{\Lambda} t}\boldsymbol{U}^{-1}\boldsymbol{n}(0)$$

Equation 9.5

where $\boldsymbol{n}(0)$ contains the initial conditions (i.e., at t = 0) for each grain (i.e., $n_i(E_i, 0)$), $\boldsymbol{U}$ is matrix of eigenvectors obtained from diagonalization of $\boldsymbol{M}$, and $\boldsymbol{\Lambda}$ is a diagonal matrix of the corresponding eigenvalues. For a conserved system (i.e., one for which the previously discussed 'infinite sink' approximation has not been introduced) with S different chemical configurations (or wells), there will be S eigenpairs that are substantially smaller in absolute magnitude (i.e.,. they are less negative) than the other eigenvalues. The first eigenvalue, often referred to as λ_0, will be equal to zero, and the corresponding eigenvector gives the equilibrium Boltzmann distributions of the different isomers on the PES.[3,5] For systems that utilize the infinite sink approximation, diagonalization of $\boldsymbol{M}$ does not yield an eigenvalue equal to zero. In both cases the S eigenvalues are often referred to as the 'chemically significant' eigenvalues. Along with their corresponding eigenvectors, they describe the time evolution of the system as it approaches equilibrium. The 'chemically significant' eigenvalues are those that determine the experimentally observed phenomenological rates measured in kinetics experiments, since they describe reaction and interconversion between the different molecular configurations on the PES.[5,14] The remaining eigenvalues – those that are much more negative than the chemically significant eigenvalues – correspond to collisional relaxation on very short time scales, and have been referred to as the internal energy relaxation eigenvalues (IERE).[2,6]

The formulation of the EGME in terms of grains essentially corresponds to expanding the solutions of Eq. 9.4 in a basis of delta functions whose maxima lie at the centre of the grain. In general, chemical problems are not concerned with the time evolution of every

single grain for a particular isomer. However, in order to obtain a convergent solution to Eq. 9.4, the grain size must be smaller than $\langle \Delta E \rangle_d$, which is generally less than ~150 cm^{-1}.

In general objective of EGME calculations is to determine quantities that can be compared to experimental observables: product yields, branching ratios, and rate coefficients, all of which may be a function of temperature and / or pressure. Extracting information about product yields and branching ratios is straightforward. The solutions to the EGME yield multi-exponential functions that describe the time dependent population in each grain, and the grain populations are normalized to unity,[5] and time dependent species concentrations are calculated by summing over the appropriate grains. Extraction of phenomenological rate coefficients from the EGME solution for comparison with experimentally determined rate coefficients is less straightforward. In general, two methods have been proposed: one by Klippenstein and Miller,[9] and one by Bartis and Widom.[2,17] In the Klippenstein and Miller approach, the rate coefficient for a particular transformation on the PES is obtained by taking the time derivative of the exponential expression that describes the $n_i(E_i, t)$ terms for a particular isomer. The Bartis and Widom approach defines a set of rate coefficients based on the eigenpairs of the chemically significant eigenvalues.

14.1.1 The Bimolecular Source Term

Provision is made within MESMER for bimolecular reaction to act as a source term, the main restriction being that one of the reactants must be in excess, so that the bimolecular association occurs under pseudo-first order conditions. A generic association reaction

$$A + B \rightleftharpoons C$$

has a forward association rate constant, k_a, and a backward dissociation rate constant, k_d. If the conditions are such that [A] >> [B], this reaction can be represented as pseudo-isomerization reaction,

$$B \rightleftharpoons C$$

where the forward rate coefficient is $k_a' = k_a A$. At equilibrium, the forward and reverse rates are equal:

$$k_a' x_B = k_d x_C$$

Equation 9.6

where x_B and x_C are the equilibrium fractions of *B* and *C*. Both k_a' and k_d are canonical rates of reaction and so depend on temperature, but each is related to their corresponding microcanonical rate constants $k(E)$. The canonical dissociation rate, k_d, is related to the microcanonical dissociation rate, $k_d(E)$, through the following relation:

$$k_d = \sum_E k_d(E) f(E)$$

Equation 9.7

where $f(E) = \rho(E) \exp(-\beta E) / Q_C$ and $Q_C = \sum_E f(E)$. Substituting Eq. 9.7 into 9.6 gives,

$$k_a' x_B = \sum_E k_d(E) f(E)\, x_C$$

Equation 9.8

where the term $f(E)x_C$ represents the equilibrium fraction in grain *E* of adduct *C*, and the $k_d(E)$in 9.8 describe transition from grains in *C* to the bimolecular source species, *B*.

In implementing the bimolecular source term, the symmetrization master equation matrix that follows detailed balance is exploited. The asymmetric ME transition matrix, $\boldsymbol{M}$, would require two sets of microcanonical rate coefficients: one that describes the transition from the bimolecular source term *B* to grains in *C*, and one that describes the transition from grains in *C* to the bimolecular source term *B*. However, for the symmetric matrix $\boldsymbol{S}$, we only need to calculate the $k_d(E)$ because the rows and columns that correspond to transition between grains in *C* and the source term *B* are identical by detailed balance. In general, matrix elements of the symmetric matrix $\boldsymbol{S}$ are related to the matrix elements of the asymmetric matrix $\boldsymbol{M}$ as follows:

$$S_{ij} = M_{ij} \left(\frac{f_j}{f_i}\right)^{1/2} = M_{ji} \left(\frac{f_i}{f_j}\right)^{1/2} = S_{ji}$$

Equation 9.9

where f_i and f_j are the equilibrium fractions in grains *i* and *j*, respectively. If a well has N grains, then the row and column that correspond to transitions involving the bimolecular source term have index N+1. The $\boldsymbol{S}$ matrix elements for transition between grains in *C* and the bimolecular source term, *B*, are calculated using Eq. 9.9 and recognizing that the M_{ij}

matrix elements are equivalent to $k_d(E)$. It follows that f_j is equivalent to the expression $f(E)x_C$ in 9.8, and f_i is equivalent to x_B. Substituting these into Eq. 9.9, we obtain the matrix elements in $\boldsymbol{S}$, which run from $E = 1$ to $E = N$:

$$S_{N+1,E} = S_{E,N+1} = k_d(E)\left(\frac{f(E)x_C}{x_B}\right)^{1/2}$$

Equation 9.10

The final matrix element, $S_{N+1,N+1}$, is simply k'_a, the pseudo first order canonical loss rate constant of the bimolecular source term to all the grains in *C*. The equilibrium constant for the above reaction, *K*, is,

$$K = \frac{k'_a}{k_d} = \frac{x_C}{x_B}$$

Equation 9.11

then k'_a may be obtained by rearranging Eq. 9.11:

$$k'_a = Kk_d = S_{N+1,N+1}$$

Equation 9.12

14.2 Other Methods for solving the master equation

14.2.1 The Reservoir State Approximation

This method assumes that significant portions of low energy molecular phase space are in a Boltzmann distribution throughout the course of the reaction. It is usually appropriate for grains which are more than a few *kT* below the lowest reaction threshold, and when the rate of collisional deactivation is faster than the rate of reaction (which is usually the case at moderate pressures). We have done extensive testing of the reservoir state approximation, and shown that it gives results nearly identical to the full ME over a range of conditions. The reservoir state approximation does not eliminate numerical problems *per se*, but it significantly truncates the size of the matrix that must be diagonalized. Since the bulk of a MESMER calculation is tied up in matrix diagonalization, the reservoir state approximation results in a far more efficient calculations (up to a factor of 30 faster!), especially in conjunction with increased precision arithmetic using the QD libraries installed with MESMER.

The reservoir state can be formulated by analogy with the treatment of the bimolecular source term. In effect, the bimolecular source term represents a collection of grains that are represented with one grain because we assume that these grains are always thermalized throughout the reaction. This saves computational effort, reducing the size of the matrix that we need to diagonalize, and it has been shown to be a good assumption so long as the reactants are thermalized throughout the course of the reaction. As long as the frequency of non reactive collisions is substantially higher than the frequency of reactive collisions, then assuming the reactants to be thermalized is a good approximation.

By analogy then, a reservoir state approximation will be good when the transition probabilities between the reservoir and the high energy grains are very small. In this case, the probability of deactivating collisions will be much greater than activating collisions. Because we are often interested in using the ME to model pressure dependence in the region of a barrier, a reservoir state that lies significantly below the barrier should not significantly affect the stochastic behaviour that happens in the region of the barrier, and which is generally the behaviour that we want to capture with a ME treatment.

The implementation of the reservoir state is very similar to that of the bimolecular source term. There are only a few differences, and the problem is slightly simplified because the process is unimolecular, so concerns about pseudo first order conditions do not arise. Using the same scheme used to describe the bimolecular source term, we imagine the following reaction:

$$B \rightleftharpoons C$$

where the forward rate constant, k_a, now represents the rate constant for activation from the reservoir state, *B*, into the active state, *C*. The backward rate constant, k_d, represents the rate constant for deactivation from the active state, *C*, into the reservoir state, *B*. At equilibrium, the forward and reverse rates are equal:

$$k_a x_B = k_d x_C$$

Equation 9.13

where x_B and x_C are the equilibrium fractions of *B* and *C*. Both k_a and k_d are canonical rates of reaction and so depend on temperature, but each is related to their corresponding microcanonical rate constants $k(E)$. The canonical deactivation rate, k_d, is related to the microcanonical dissociation rates, $k_d(E)$, through the following relation:

$$k_d = \sum_E k_d(E) f(E)$$

Equation 9.14

Where, again, $f(E) = \rho(E)\exp(-\beta E)/Q_C$ and $Q_C = \sum_E f(E)$.. Now, the most significant difference between treating the bimolecular source term and the reservoir state is how we treat $k_d(E)$. In our treatment of the bimolecular source term, $k_d(E)$ was obtained from either ILT or RRKM methods. But we cannot use these techniques for treating $k_d(E)$ into the reservoir state. The $k_d(E)$ in this case correspond to the energy dependent rates at which species in grains within *C* are deactivated into the state *B*, and the most straightforward way to do this is to use the downward collision transition probabilities. To get $k_d(E)$ for deactivation from a grain in *C* into the reservoir B, we must sum the normalized downward transition probabilities, $P(i|E)$, for deactivation of a particular grain in *C* into *every possible grain in B*. If *B* spans the energy range from E_0 to E_t and *C* spans the energy range from E_{t+1} to E_∞, then $k_d(E)$ for deactivation of a grain *E* in *C* to the reservoir state B may be calculated as follows:

$$k_d(E) = \omega \sum_{i=E_0}^{i=E_t} P(i|E)$$

Equation 9.15

where ω is the collision frequency. Subsituting Eq. 9.14 into 9.13, we obtain:

$$k_a x_B = \sum_E k_d(E) f(E)\, x_C$$

Equation 9.16

where the term $f(E)x_C$ represents the equilibrium fraction in grain *E* of the active state *C*, and $k_d(E)$ are calculated according to Eq. 9.15. Similar to the case of the bimolecular source term, the asymmetric ME transition matrix, $\boldsymbol{M}$, requires microcanonical rate coefficients that describe the transition from the reservoir *B* to the grains in *C*. In practice though, we only need to calculate $k_d(E)$ because the symmetrized ME matrix, $\boldsymbol{S}$, needs to have reservoir row and column vectors that are identical by detailed balance (see Eq. 8.9). Let's say that an active state, *C*, has N grains, so that the row and column that correspond to transitions with the reservoir state have index N+1. Recognizing that the M_{ij} matrix elements are equivalent to $k_d(E)$, then f_j is equivalent to $f(E)x_C$ and f_i is equivalent to x_B. Substituting these into Eq. 9.9, we obtain the matrix elements in $\boldsymbol{S}$ for the reservoir state, which run from $E = 1$ to $E = N$:

$$S_{N+1,E} = S_{E,N+1} = k_d(E)\left(\frac{f(E)x_C}{x_B}\right)^{1/2}$$

Equation 9.17

The final matrix element, $S_{N+1,N+1}$, is simply k_a, the canonical loss rate constant of the reservoir state to all the activated grains in *C*. If the (R3) equilibrium constant, *K*, is as follows:

$$K = \frac{k_a}{k_d} = \frac{x_C}{x_B}$$

Equation 9.18

then k_a may be obtained by rearranging Eq. 9.18:

$$k_a = Kk_d = S_{N+1,N+1}$$

Equation 9.19

14.2.2 The Contracted Basis Set Approach

The aim of this approach is to reduce the size of the final matrix to be diagonalized by using a basis set that is constructed from the diagonalization of the individual isomer collision energy transfer operators. It is similar in spirit to the basis set methods reported by Venkatash *et al*[15,16]. Consider a two well isomerization system, after symmetrization the overall master equation matrix can be written as:

$$\boldsymbol{M} = \begin{pmatrix} \omega(\boldsymbol{S}_A - \mathbf{1}) & \mathbf{0} \\ \mathbf{0} & \omega(\boldsymbol{S}_B - \mathbf{1}) \end{pmatrix} + \begin{pmatrix} -\boldsymbol{K}_A & \boldsymbol{K}' \\ \boldsymbol{K}' & -\boldsymbol{K}_B \end{pmatrix}$$

Equation 9.20

where the first term on the right hand side represents the collisional activation/deactivation process and the second represents reactive exchange. The first term on the right hand side is clearly block diagonal and each of these blocks can be diagonalized independently, generating two sets of eigenvectors which are orthogonal within each set and between each set. If these sets are denoted $\boldsymbol{U}_A$ and $\boldsymbol{U}_B$, then the combined eigenvector matrix for the first term on the right hand side can be written as:

$$\boldsymbol{U} = \begin{pmatrix} \boldsymbol{U}_A & \mathbf{0} \\ \mathbf{0} & \boldsymbol{U}_B \end{pmatrix}$$

Equation 9.21

This matrix can then be applied as a similarity transform to give:

$$\boldsymbol{M' = UMU^T} = \begin{pmatrix} \boldsymbol{\Lambda_A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Lambda_B} \end{pmatrix} + \begin{pmatrix} \boldsymbol{U_A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{U_B} \end{pmatrix} \begin{pmatrix} \boldsymbol{-K_A} & \boldsymbol{K'} \\ \boldsymbol{K'} & \boldsymbol{-K_B} \end{pmatrix} \begin{pmatrix} \boldsymbol{U_A^T} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{U_B^T} \end{pmatrix}$$

Equation 9.22

While all the eigenvectors have been used in the above, this is not always necessary to converge the smallest eigenvalues of the overall matrix. If a smaller set is sufficient then there is the potential for increased computational efficiency.

14.3 Inverse Laplace Transform

As discussed above the inverse Laplace transform (ILT) can be used to obtain microcanonical rate coefficients from an existing Arrhenius form. In the following an outline is given of how the methods are applied for both the case where the Arrhenius form is for a unimolecular dissociation and for a bimolecular association.

14.3.1 Unimolecular ILT

The canonical high pressure rate coefficient may be expressed as:

$$k(\beta) = \frac{1}{Q(\beta)} \int_0^\infty k(E)\rho(E) \exp(-\beta E)\, dE$$

Equation 9.23

Where $\rho(E)$ is the rovibrational density of states and $Q(\beta)$ is the corresponding canonical partition function. Substituting Eq. 9.2 for $k(E)$ gives,

$$k(\beta) = \frac{1}{hQ(\beta)} \int_0^\infty W(E) \exp(-\beta E)\, dE$$

Equation 9.24

and rearranging gives:

$$hQ(\beta)k(\beta) = \int_0^\infty W(E) \exp(-\beta E)\, dE = \mathcal{L}[W(E)]$$

Equation 9.25

$W(E)$ can be recovered by inverting the Laplace transform,

$$W(E) = h\mathcal{L}^{-1}[Q(\beta)k(\beta)]$$

Equation 9.26

If $k(\beta)$ can be represented by the modified Arrhenius expression:

$$k(\beta) = A_0\left(\frac{\beta_0}{\beta}\right)^n \exp(-\beta E_a)$$

Equation 9.27

it follows that:

$$\frac{W(E)}{h} = k(E)\rho(E) = A_0\beta_0^n\mathcal{L}^{-1}\left[\frac{Q(\beta)}{\beta^n}\exp(-\beta E_a)\right]$$

Equation 9.28

Further progress can be made by applying the convolution theorem:

$$\mathcal{L}^{-1}[Q(\beta)G(\beta)] = q \otimes g$$

Equation 9.29

where Q and q, and G and g are transform pairs, and $\otimes$ denotes convolution. Solution of 9.28 is possible by recognizing that

$$\mathcal{L}^{-1}[Q(\beta)] = \rho(E)$$

Equation 9.30

And

$$\mathcal{L}^{-1}[G(\beta)] = \mathcal{L}^{-1}\left[\frac{1}{\beta^n}\exp(-\beta E_a)\right] = \frac{(E-E_a)^{n-1}u(E-E_a)}{\Gamma(n)}$$

Equation 9.31

where $u(E-E_a) = 0$ if $E < E_a$ and is unity otherwise, is the Heavyside step function. Convolving the ILTs in Eq. 9.30 and 9.31 gives:

$$k(E)\rho(E) = \frac{A_0\beta_0^n}{\Gamma(n)}\int_0^E d\tau\rho(E-\tau)(\tau-E_a)^{n-1}u(\tau-E_a)$$

Equation 9.32

The units should be input follows:

$\rho(E)$ in states per cm^{-1}

A_0 in molecules cm^{-3} s^{-1}

$\beta_0 = \frac{1}{k_B T_0}$ where k_B is in units of cm^{-1} K^{-1}, and T_0 in units of K.

14.3.2 The association ILT

A similar expression can be obtained for the case where the Arrhenius expression is for the high pressure association rate coefficient. Under high pressure conditions the forward and the reverse rate coefficients are related by the equilibrium constant as indicated by Eq. 9.11:

$$k_d(\beta) = K_e(\beta)k_a(\beta)$$

Equation 9.33

If $k_a(\beta)$ has a modified Arrhenius form then the equivalent expression to Eq. 9.28 is,

$$\frac{W(E)}{h} = k(E)\rho(E) = A_0\beta_0^n\mathcal{L}^{-1}\left[K_e(\beta)\frac{Q(\beta)}{\beta^n}\exp(-\beta E_a)\right]$$

Equation 9.34

Solution of Eq. 9.34 is complicated by the appearance of translational degrees of freedom in the equilibrium constant, but otherwise proceeds as previously by exploiting the convolution theorem. The final result is (see ref. 8),

$$k(E)\rho(E) = \frac{A_0\beta_0^n}{\Gamma(n+1.5)}\left(\frac{2\pi\mu}{h^2}\right)^{3/2}\left(\frac{g_A g_B}{g_C}\right)\int_0^E d\tau\rho_R(E-\tau)(\tau-E_a-\Delta H_0^0)^{n+0.5}u(\tau-E_a-\Delta H_0^0)$$

Equation 9.35

where $\rho_R(E)$ is the convolved density of states for the associating pair, ΔH_0^0 is the enthalpy of reaction, μ is the reduced mass of the system and g_X is the spin degeneracy of species X.

14.3.3 The C' constant in MESMER ILT

The constant C' that occurs in the MESMER implementation of ILT follows from the translational partition function,

$$Q_t = \left(\frac{2\pi mkT}{h^2}\right)^{3/2} V$$

Equation 9.36

where all quantities are in standard SI units. For ease of computation, it is useful to re-write Q_t in terms of the molar mass M (g/mol) and the reciprocal temperature β' expressed in wave numbers:

$$m = M/10^3 L$$

Equation 9.37

β'= hc/kT

Equation 9.38

where L is Avogadro's number and c is the speed of light expressed in cm/s, otherwise all quantities are in SI units. Inserting Eqs. 9.37 and 9.38 into 9.36 gives:

$$Q_t = \left(\frac{2\pi Mc}{10^3 L\beta' h}\right)^{3/2} V = C\left(\frac{M}{\beta'}\right)^{3/2} V$$

Equation 9.39

where C is given by

$$C = \left(\frac{2\pi c}{10^3 Lh}\right)^{3/2}$$

Equation 9.40

Volume is more conveniently expressed in cm^3 and to account for this Eq. 9.40 can be written as

$$Q_t = C'\left(\frac{M}{\beta'}\right)^{3/2} V'$$

Equation 9.41

where $V'$is the volume now expressed in cm^3 and the constant $C'$is given by,

$$C' = \left(\frac{2\pi c}{10^3 Lh}\right)^{3/2} 10^{-6} = \left(\frac{2\pi c}{10^7 Lh}\right)^{3/2}$$

Equation 9.42

Substituting in $L = 6.02205 \times 10^{23}$, $h = 6.62618 \times 10^{-34}$Js and $c = 2.997925 \times 10^{10}$cm/s gives $C' = 3.2433 \times 10^{20}$ (mol/g/cm)$^{3/2}$. This is the value that is defined in the `constant.h` file of the MESMER source code.

15 References

[1] Miller, W. H. Tunneling corrections to unimolecular rate constants, with application to formaldehyde, *Journal of the American Chemical Society*, 101, 6810-6814, 1979.

[2] Robertson, S. H., Pilling, M. J., Jitariu, L. C., and Hillier, I. H. Master equation methods for multiple well systems: application to the 1-,2-pentyl system, *Phys Chem Chem Phys* 9, 4085-4097, 2007.

[3] Pilling, M. J., and Robertson, S. H. Master equation models for chemical reactions of importance in combustion, *Annual Review of Physical Chemistry*, 54, 245-275, 2003.

[4] Holbrook, K. A., Pilling, M. J., and Robertson, S. H. *Unimolecular Reactions*, 2nd ed., John Wiley & Sons, Chichester, 223 pp., 1996.

[5] Klippenstein, S. J., and Miller, J. A. From the Time-Dependent, Multiple-Well Master Equation to Phenomenological Rate Coefficients, *J Phys Chem A*, 106, 9267-9277, 2002.

[6] Miller, J. A., and Klippenstein, S. J. Master Equation Methods in Gas Phase Chemical Kinetics, *J Phys Chem A*, 110, 10528-10544, 2006.

[7] Gannon, K. L., Glowacki, D. R., Blitz, M. A., Hughes, K. J., Pilling, M. J., and Seakins, P. W. H Atom Yields from the Reactions of CN Radicals with C_2H_2, C_2H_4, C_3H_6, trans-2-C_4H_8, and iso-C_4H_8, *J Phys Chem A*, 111, 6679-6692, 2007.

[8] Davies, J. W., Green, N. J. B., and Pilling, M. J. The testing of models for unimolecular decomposition via inverse Laplace transformation of experimental recombination rate data, *Chemical Physics Letters*, 126, 373-379, 1986.

[9] Miller, J. A., Klippenstein, S. J., and Raffy, C. Solution of Some One- and Two-Dimensional Master Equation Models for Thermal Dissociation: The Dissociation of Methane in the Low-Pressure Limit, *J Phys Chem A*, 106, 4904-4913, 2002.

[10] Miller, J. A. Concluding Remarks, *Faraday Discussions*, 119, 461-475, 2001.

[11] Gilbert, R. G., and Smith, S. C. *Theory of Unimolecular and Recombination Reactions*, Blackwell Scientific Publications, Oxford, 1990.

[12] Baer, T., and Hase, W. L. *Unimolecular reaction dynamics: theory and experiments*, Oxford University Press, New York, 324-368 pp., 1996.

[13] Robertson, S. H., Pilling, M. J., Baulch, D. L., and Green, N. J. B. Fitting of Pressure-Dependent Kinetic Rate Data by Master Equation Inverse Laplace Transform Analysis, *J. Phys. Chem.*, 99, 13452-13460, 1995.

[14] Green, N. J. B., and Bhatti, Z. A. Steady-state master equation methods, *PCCP*, 9, 4275-4290, 2007.

[15] Venkatesh, P. K., Dean, A. M., Cohen, M. H., and Carr, R. W. Master equation analysis of intermolecular energy transfer in multiple-well, multiple-channel unimolecular reactions. II. Numerical methods and application to the mechanism of the $C_2H_5+O_2$ reaction, *Journal of Chemical Physics*, 111, 8313-8329, 1999.

[16] Venkatesh, P. K., Dean, A. M., Cohen, M. H., and Carr, R. W. Master equation analysis of intermolecular energy transfer in multiple-well, multiple-channel unimolecular reactions. I. Basic theory, *Journal of Chemical Physics*, 107, 8904-8916, 1997.

[17] Bartis, J. T., and Widom, B. Stochastic models of the interconversion of three or more chemical species, *Journal of Chemical Physics*, 60, 3474-3482, 1974.