MESMER

Master Equation Solver for Multi-Energy Well Reactions

Version 7.0

User's Manual

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

Michael J. Pilling, Robin Shannon, Mark Blitz, Alison Tomlin, and Paul W. Seakins

Last updated: 13 November 2022

1   Acknowledgements and Citation

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

If you published results using MESMER, we would appreciate very much if you would cite the following paper:

D.R. Glowacki, C. H. Liang, C. Morley, M.J. Pilling and S.H. Robertson, "MESMER: an open-source master equation solver for multi-energy well reactions", *Journal of Physical Chemistry A*, 9545−9560, **116** (2012)

2 Notices

MESMER, Copyright (C) 2009-2021 by Struan H. Robertson, Chris Morley, David R. Glowacki, Chi-Hsiu Liang, Michael J. Pilling and contributors, is distributed under the terms of the GNU Public License version 2 (GPLv2). Details can be found at http://www.gnu.org/licenses/.

MESMER employs the QD extended precision library which is distributed under the following terms:

Copyright (c) 2003, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from U.S. Dept. of Energy)

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

(2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

(3) Neither the name of Lawrence Berkeley National Laboratory, U.S. Dept. of Energy nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR

MESMER employs the TinyXML library which is distributed under the terms of the zlib license.

3   Contents

6

4 What's New in MESMER 7.0

New features, input changes and bug fixes:

- Radiation transitions. A radiation transition model has been added that assumes a Planck distribution with same temperature as the bath gas (See section).

- Excess species concentration can now defined as a fraction of the bath gas concentration (see sections 7.3.2 and 8.3.1). We thank Timo Pekkanen of the University of Helsinki, for suggesting this change.

- The function of the keyword `refReaction` has been extended to allow better reporting of local sink reactions (see section 8.3.2).

- Energy transfer – activated energy transfer model.

- Energy transfer – linear dependence on initial energy.

- Improved reporting of bimolecular rate coefficients.

- Symmetry number added to `FourierGrid1D` method to account for inversion symmetry and similar (See section 11.2.3). We thank Timo Pekkanen of the University of Helsinki, for suggesting this change.

- Fourier coefficient analysis of numerical hindering potentials has been extended to include data that is not evenly spaced (See section). We thank Tom Stephenson of … College, US, for suggested this change.

5   Introduction

The modelling of unimolecular systems has applications 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, experimental techniques may dictate that reactions relevant to the atmosphere need to be performed at low pressures (5 - 50 mbar); however, their relevance is at higher pressures, so the experimental results must be extrapolated to the pressures characteristic of the lower atmosphere (200 - 1000 mbar). 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, see Figure 1, 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. Typical systems treated using MESMER also include a bimolecular source and the formation of products by dissociation from one or more of the wells.

Figure 1: Schematic of the processes involved in MESMER. (Taken from D.R. Glowacki, C.-H. Liang, C. Morley, M.J. Pilling and S.H. Robertson, *Journal of Physical Chemistry A*, 9545−9560, **116** (2012)) [1]

MESMER has been written to offer a flexible approach to ME treatments of such complex systems. We have also attempted to incorporate various facilities that make it easy to apply the ME to gas kinetics. 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 Accessing, 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, MESMER is developed on Windows. For large production runs, where several hundreds or even thousands of ME calculations are required to explore the sensitivity of the results over the model parameter space, LINUX may be a better option.

6.1 Accessing MESMER

MESMER is hosted by the 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.

6.2 Parallel MESMER

As of MESMER 5.0 there is support for parallel execution. The parallel substrate on which parallel MESMER is built is MPI. This substrate was selected because of the nature of calculations for which MESMER is often used. Typically, calculations such as fitting often involve the repetitive calculation of rate coefficients for many different sets of conditions. Such repetitive calculations will usually involve the construction and diagonalization of a number of ME transition matrices, one for each condition, and as such these operations can be regarded as individual processes. It is this feature that is exploited in the parallel implementation in MESMER, the MPI substrate is used to construct and diagonalize transition matrices for different conditions in separate MPI ranks, the results being drawn together on the master rank.

6.3 Windows

6.3.1 Installing the Binary on Windows

Download the Windows installer and execute it. It will ask you to accept the GPLv2 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, go to Settings/Apps and select MESMER and click Uninstall.

Note. It is recommended that you uninstall any earlier version of MESMER that is installed before installing the latest version.

6.3.2 Compiling 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++ 2019/2022 integrated development environment. Building and developing MESMER can be done using either the free Visual C++ 2022 Community Edition or the full version of Visual Studio (which isn't free).

To build MESMER using VS2022, you could download the MESMER tar.gz distribution as described in section 6.4, 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_4.1

To build it, go to the `\Windows VC17` folder, and use Visual C++ to open the file `MESMER.sln`. Clicking on the VS2019 command `Build the Solution` will build the binary executable in `\Windows VC17\Mesmer` folder. (Depending on the MESMER release that you download, you may see a `\Windows VC12`, `\Windows VC14` or

13

`\Windows VC17` folders. These folders contain the projects and solution files for VS2012, VS2015 and VS2017 respectively.)

6.3.3 Compiling Parallel MESMER on Windows

In order to assist the building of parallel MESMER a configuration has been created in the solutions for both VS2017 and VS2019 called "MPIRelease" which some pre-set values to guide you through the build process and it is recommended that you select this configuration using the configuration manager. The procedure for building parallel MESMER on Windows largely follows the procedure above, the main difference being that an MPI environment needs to be available. It is recommended that "Microsoft MPI" is used. The pre-compiled Windows executable has been created using version 10 of MS-MPI. When downloading the MS-MPI v10 be sure to download both msmpisdk.msi (contains headers and libraries) and MSMpiSetup.exe (contains the MPI run time environment). Install both of these packages as described in the associated documentation. When this is complete the following additional changes, in the MESMER project property pages are required in order to build parallel MESMER (if you are using the "MPIRelease" configuration with standard locations these steps will have already been completed):

1. In the C/C++/General section, update the "Additional Include Directories" to include the path to the `…\Microsoft SDKs\MPI\Include` folder.
2. In the C/C++/pre-processor section, update the "Pre-processor" definitions to include "PARALLEL".
3. In the Linker/General section, update the "Additional Library Directories" to include the path to the `…\Microsoft SDKs\MPI\lib\x86` folder.
4. In the Linker/Input section add the libraries: msmpi.lib, msmpifec.lib and msmpifmc.lib.

6.3.4 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 6.6 for a more complete description of the options and syntax of the command line.

Alternatively, if the user is not familiar with using the command line, MESMER may be executed through right clicking on the desired xml input file and selecting the "Open with Mesmer" option. The additional command line options mentioned above are not available if MESMER is executed in this way.

6.3.5 Running Parallel MESMER on Windows

Running parallel MESMER on Windows is similar to the above, but it is necessary to have a parallel environment in place. It is recommended that "Microsoft MPI" is used, and the user is referred to the instructions at the being of section 6.3.3 for details on how to installed this. Once installed parallel MESMER can be run from the command line, for example to run the `ipropyl_LM.xml` example in the `examples/i-propyl` folder, on four processors the following command should be invoked:

```
<MPIInstalPath>\mpiexec -n 4 mesmer.exe -N ipropyl_LM.xml
```

6.4 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.

The fasted way to build MESMER on Linux is to use the shell script `Build.sh` that is in the top-level directory of a Linux installation. This will build the libraries that MESMER depends on as well as MESMER itself. However, there are occasions when more control over the compilation process is needed and this is described in the following sections. 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.

6.4.1   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
```

6.4.2   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 deep wells, low temperatures, and low pressures. To address this issue MESMER calculations can be performed using significantly increased precision available in the QD libraries written by Yozo Hida *et al*, Software Directory (davidhbailey.com). 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 --prefix $PWD/src

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.

Calculations using extended precession are more expensive. For some systems the increase in computational cost can be offset by using Reservoir states see section 7.3.1.

6.4.3   Compiling 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.

6.4.4 Compiling Parallel MESMER on Linux

Before compiling MESMER to run in parallel on Linux, you must ensure that an MPI environment has been installed on the machine that you intend to run MESMER on. If you are unsure about this, it may be best to seek the advice of your systems administrator. There are a number of MPI implementations, such as MPICH (https://www.mpich.org/) or OpenMPI (https://www.open-mpi.org/), as well as commercial offerings.

Compilation of parallel MESMER is similar to described above in section 6.4.3: both the TinyXML and QD libraries should be built in the same way, however change to the `Makefile` in the `/src` directory are required. Changes are needed in order to locate MPI header files and libraries: The included path variable (line 108 of `Makefile`), `INCS`, needs to be extended to include the location of the `mpi.h` file,

```
INCS := -I../tinyxml/ -I../qd/include -I<MPIInstalPath>/include
```

where `<MPI install path>` is the location of the MPI installation. Similarly, the library path variable (line 108 of `Makefile`), `INCS`, needs to be extended to include location of the MPI libraries,

```
LIBS := ../tinyxml/tinyxml.a ../qd/src/lib/libqd.a <MPIInstalPath>/libs
```

If the MPI installation includes *.a files in the libs folder, it may be necessary to specify /libs/*.so in the above path to avoid errors. Once these changes have been made, the executable can be made by invoking the command,

```
make install DEBUG=NO PARALLEL=YES
```

Alternatively, starting from a fresh download, make the changes to the `Makefile` just described, and run the script `BuildParallel.sh` which will build all the libraries and create the MPI executable.

6.4.5   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.

6.4.6 Running Parallel MESMER on Linux/UNIX/Mac

Running parallel MESMER on Linux is similar to the above, but it is necessary to have a parallel environment in place, as discussed in section 6.4.4. Once installed parallel MESMER can be run from the command line, for example to run the `ipropyl_LM.xml` example in the `examples/i-propyl` folder, on four processors the following command should be invoked:

`<MPIInstalPath>/bin/mpirun -n 4 mesmer.exe -N ipropyl_LM.xml`

6.5 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 (less than 0.1%) 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 10.1. 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.

6.6 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. The possible command line options are:

- `-?`: Writes a complete list of options.
- `-o <output filename>`: Allows the specification of the .xml output filename. If -o has no output filename, stdout is used.
- `-n`: Allows the input file to be overwritten when there is no -o option, otherwise the output file is mesmer_out.xml. Any file about to be overwritten is saved with suffix _prev to its name.
- `-N`: Use the input filename for .test and .log, default is mesmer.test and mesmer.log.
- `-p`: Parse the input file only - no calculation. Useful for checking the input file content.
- `-q`: Run a QA test. (Compares mesmer.test with a definitive version.)
- `-w#`: Display only warning messages that are at least as severe as:
    - o   0 No messages
    - o   1 Errors
    - o   2 Warnings
    - o   3 Information
- `-t`: Output a table of available plugins.
- `-T`: As -t but with descriptions, if available.
- `-V`: Output Mesmer version.

Hence 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.

6.7 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`

7   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. The MESMER input structure draws heavily on the CML (Chemical Markup Language) format. 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 output XML file can be used as an input file.

The files can contain data in excess of that required by MESMER. For instance, they may contain chemical structure information, which may not be 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.

7.1   Editing and Viewing Data Files

For users unfamiliar with XML syntax, a web based GUI (graphical user interface) has been developed by Dr Xiaoqing You and colleagues. A tutorial for getting started with this is included in the documentation folder. The GUI is located at:

http://www.mesmergui.cn/

For users 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 Visual Studio, 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 (version 3 or later), as shown in Figure 2:. 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 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. The above procedure needs to be done only once.
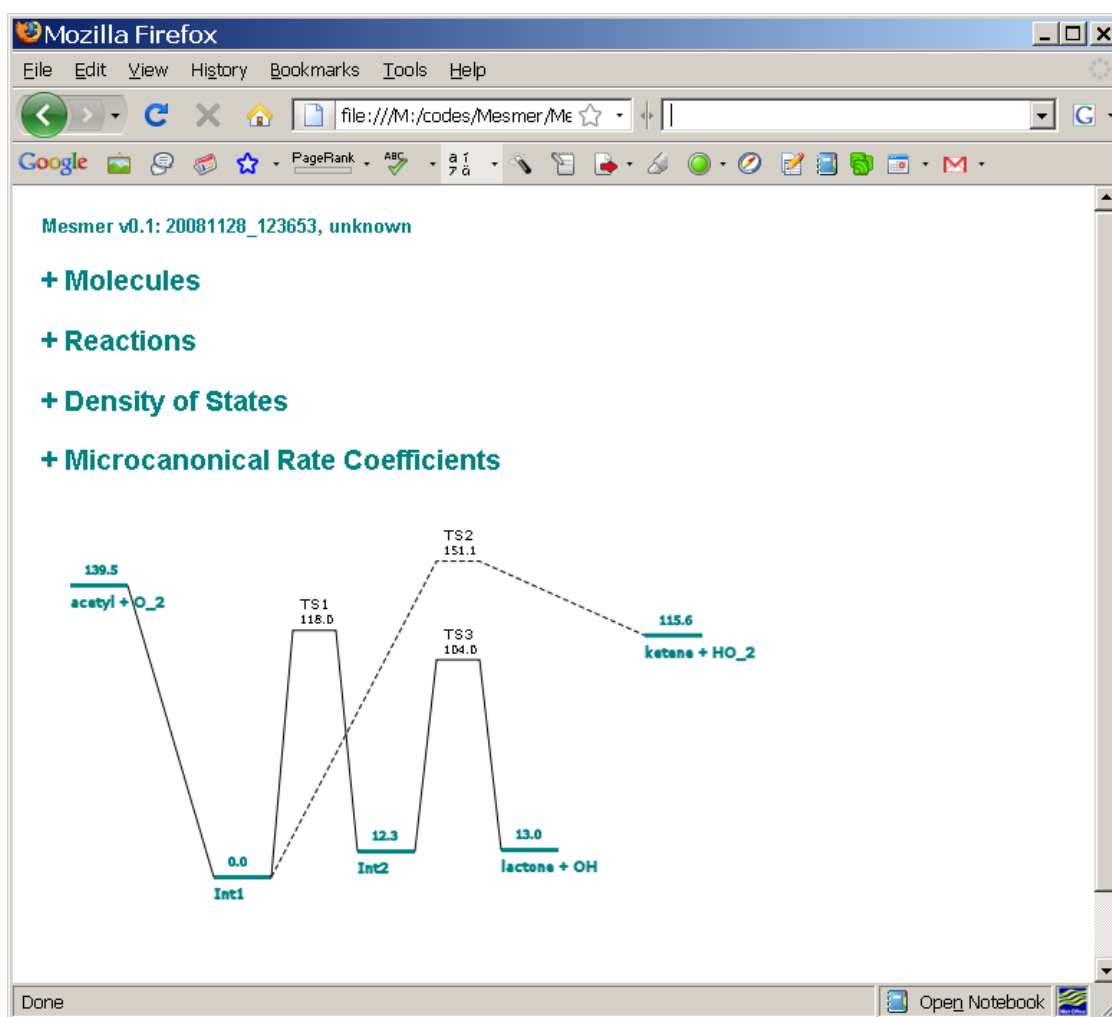
Figure 2: Viewing the potential energy surface of Acetyl (CH$_3$CO)+O$_2$ reaction by opening the XML input file in Firefox 3.

This modification does reduce, to some extent, the security of the browser and you may feel it is worthwhile to create a separate Firefox instance with an unmodified browser profile for normal web access.

7.2 Validation of MESMER input

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

In addition, as of MESMER 3.0, an XML schema has been introduced that allows input files to be validated. In loose terms, an XML schema is a set of rules, over and above the basic XML syntax, that defines the structure, type and relation of the elements and their attributes must have if it is to conform to a given document type, in the present case the type being a MESMER input file. XML schema can be expressed in a number of ways, and .xsd or "XML Schema Definition" files have been chosen to implement the MESMER schema. The MESMER schema can be used in conjunction with Visual Studio 2010 (and later) to check the content of input files as follows:

1. Start Visual Studio and open the MESMER input file of interest.
2. On the main ribbon, next to the File, Edit and Build tools, an item "XML" will appear. Click on this item and select "Schemas…".
3. A dialogue displaying the schemas that are currently being applied will be displayed.
4. Click "Add…", a browse dialogue will be displayed. Navigate to the top level of your MESMER installation and from there navigate to the folder called "schemas".
5. A number of .xsd files will be displayed. Select all these files. They will appear in the schema list.
6. Click "ok" on the schema dialogue and return to the input XML input file.
7. Any elements that do not conform to the schema will be indicated by blue zig-zag underlining. Hovering over this underlining will cause a dialogue box to be displayed giving more details about the possible problem that has been detected. Badly formed XML is indicated by red zig-zag underlining

7.3   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 XML is almost entirely in two namespaces: MESMER, shown by a `me:` prefix on element names; and CML, which as the default namespace has elements with no prefix. As described in section 8.2, the XML file structure was altered slight at Mesmer 3.0. The description here is for the new version but includes some references to the old. (MESMER will read both types of file.)

The MESMER data file has a top level element `<me: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. There can be multiple condition sections

(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 There can be multiple control sections.

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

7.3.1 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. It preferably should start with a letter and contain only letters, digits, and -_() characters , and no spaces;

(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) `atomarray` and `bondarray` containing, respectively, several atoms and bonds. These and other CML elements describing the chemical structure are not usually necessary for the calculation to proceed, but may be used, for example in hindered rotor calculations, or if the molecular weight has not been explicitly specified;

(4) `spinMultiplicity` is an attribute on `molecule`;

(5) A `propertyList` array, the elements of which are `property` elements of a `molecule`. Each `property` element has a `dictRef` attribute which specifies what type of `property` is subsequently defined. A `property` type that requires only a single number, such as zero-point energy location or a symmetry number, is specified to be `scalar`. A `property` type that requires a list, such as vibrational frequencies and rotational constants, is specified in an `array`. A `property` type that represents a quantity that is a matrix such as a Hessian, is specified to be `matrix`. For `property` types that 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 1

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.

Another property that requires further information is the `me:hessian` property. This property allows a Hessian to be read which can be analysed to determine the vibrational frequencies instead of entering them using `me:vibFreqs` or when the elimination of frequencies due to internal rotational is required (see section 11.2.3). A typical specification might be:

```
<matrix matrixType="squareSymmetricLT" rows="42" units="kJ/mol/Ang2">
```

The keyword `matrixType` specifies the matrix type and how it is stored. In the case of a Hessian it must be symmetric, but can be specified by either by the lower triangular elements (`squareSymmetricLT`) or the upper triangular elements (`squareSymmetricUT`), in both cases including the diagonal.

The `me:ZPE` property, and the related properties, `me:Hf0`, `me:HfAT0`, `me:Hf298`, specify the location of potential energy surface features. Discussion of these important keywords is deferred to section 7.3.1.1.

The important properties `me:epsilon` and `me:sigma` are the effective Lennard-Jones parameters for the interaction of a target species with a bath gas molecule and are used in calculating the collision frequency. Ideally these numbers would be determined in an independent measurement and such experimental parameters should used in preference to any others. However, experimental parameters are relatively rare and are almost always unknown for radical species. Typically, the Lennard-Jones for radical species can be equated with those of a parent molecule, if known. If the parameters are unknown, MESMER will attempt to calculate a set of parameters based on the prescription described by Jasper.[2] This prescription calculates collision parameters for hydrocarbons, alcohols, and hydroperoxides based on the number of heavy atoms in the target species. The identity of the bath gas species with which the target molecule can collide, and therefore the combined Lennard-Jones parameters can be calculated, are limited as follows: for hydrocarbons, Lennard-Jones parameters can be calculated for He, Ar, H_2 and N_2; for alcohols and hydroperoxides, Lennard-Jones parameters can be calculated for He, Ar. Radicals formed by the abstraction of

an H atom will have the same Lennard-Jones parameters as the parent molecule. This method of calculating Lennard-Jones parameters is invoked in the absence of any explicit Lennard-Jones parameters. If this fails, for example because the elemental composition includes elements other than C, H and O or a bath gas other than those specified above is used, then default parameters will be used. These default parameters are specified in the `defaults.xml` file and can be altered, see section 9.1.4 for more details on defaults. An example of automatically calculated parameters can be found in, `…\examples\Butan-1-ol\Butoxy_H_to_Butan-1-ol.xml`.

Details of the molecular properties are given in table 1. Some properties can be expressed in different units, and this are given in table 1, properties which must be entered in specific units are marked as "fixed".

dictRef value	Description	scalar or array	Available input units	Inserted from defaults.xml if absent?
me:ZPE	See Section 7.3.1.1	Scalar	kJ/mol or kJ per mol cm-1 or wavenumber kcal/mol or kcal per mol Hartree or au	No
me:Hf0	See Section 7.3.1.1	Scalar	as above	No
me:HfAT0	See Section 7.3.1.1	Scalar	as above	No
Me:Hf298	See Section 7.3.1.1	Scalar	as above	No
me:rotConsts	Rotational constants.	Array	cm-1, GHz or amuA^2	No
me:symmetryNumber	Rotational symmetry number.	Scalar	No units.	Yes (1)
me:TSOpticalSymmetryNumber	Transition state optical symmetry number.	Scalar	No units.	Yes (1)
me:frequenciesScaleFactor	Ab initio frequency scalar factor	Scalar	No units.	Yes (1.0)
me:vibFreqs	Harmonic vibrational frequencies	Array	cm-1	No
me:MW	Molecular weight.	Scalar	Amu	No
me:spinMultiplicity	Electron spin multiplicity. A spinMultiplicity attribute on \<molecule> is the preferred alternative.	Scalar	No units.	Yes (1)
me:epsilon	Lennard-Jones potential ε parameter (used to calculate collision frequency).	Scalar	K (fixed)	Yes(50)
me:sigma	Lennard-Jones potential σ parameter (used to calculate collision frequency).	Scalar	Å (fixed)	Yes(5)
me:deltaEDown	$\langle \Delta E \rangle_d$, the average energy transferred in a deactivating collision.	Scalar	cm^{-1}(fixed)	Yes(130 with warning)
me:hessian	Molecular Hessian matrix.	Matrix	kJ/mol/Ang2 or kcal/mol/Ang2 or Hartree/Bohr2	No

Table 1: 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. Those that are automatically given a default value, if absent, are shown. The values shown are those initially in defaults.xml, but are editable by the user and may have been changed.

(6) me:DistributionCalcMethod, which specifies what the initial distribution is in a given well. More details of the types of distribution and their use are given in section 11.2.6.

(7) 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>
```

(8) `me:DOSCMethod`, which specifies the principal method for MESMER to use in calculating external rotational states density of states. For active species one, and only one, `me:DOSCMethod` method must be specified. Classical and quantum mechanical methods for calculating the external rotational density of states are available, and they are specified with the respective text `ClassicalRotors` and `QMRotors`, e.g. `<me:DOSCMethod>QMRotors </me:DOSCMethod>` or, alternatively, `<me:DOSCMethod name="QMRotors"/>`. Table 2 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. (Note, the criterion for a near symmetric top is that the modules of the asymmetric parameter, κ, should be within 5% of unity.) These methods also account for electronic degeneracy. Occasionally there are circumstances where rotational and electronic states cannot be easily decoupled 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 and the `DefinedStatesRotors` method caters for this situation. This class 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. Another situation where coupling between motions can have a significant impact on the density of states is when an internal rotor couples with the overall rotation of a species. The `ClassicalCoupledRotors` method allows the specification of a number of internal rotors which are coupled to external rotation. The detail of this method is discussed in section 11.2.3. If no `me:DOSCMethod` method is defined then, by default, `ClassicalRotors` is added. 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.

(9) `me:ExtraDOSCMethod` This tag is used to add additional density of states calculation methods. 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 in section 11.2.3. Note that, in adding such a method, proper account must be made of the number of degrees of freedom. MESMER will check if the number of degrees of freedom are correct and halt if it detects an error.

Type of rotor	MESMER recognition criteria	Rotor type	DOS or rotational energy and degeneracy expressions
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	$\rho(E) = \dfrac{1}{\sigma B}$
3d classical rotor	three rotational constants provided in the input	Classical	$\rho(E) = \dfrac{1}{\sigma}\sqrt{\dfrac{4E}{ABC}}$
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 $A = B > C$ $E_r(J,K)=BJ(J+1)+(C\text{-}B)K^2$ $J=0,1,2;K=0,\pm1,\pm2,\ldots,\pm J$ $g_{JK} = 2J+1$
Prolate or near prolate symmetric top	$I_A < I_B = I_C$	QM	For $A > B = C$ $E_r(J,K)=BJ(J+1)+(A\text{-}B)K^2$ $J=0,1,2;K=0,\pm1,\pm2,\ldots,\pm J$ $g_{JK} = 2J+1$
Asymmetric top (King *et al.* [3])	$I_A < I_B < I_C$	QM	For $A > B > C$ $E_r(J)=(A+C)J(J+1)/2+\varepsilon(\kappa)$ $g_{JK} = 2J+1$

Table 2: Methods for calculating rotational densities of sates, 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.

7.3.1.1 Potential Energy Surface (Zero Point Energy Convention)

Within MESMER all potential energy surface (PES) features of the reaction are located with respect to each other on the basis of their zero-point energy (ZPE). Figure 3 shows a schematic of a PES, with a number of measures. The values stored by MESMER are ΔH_0 (Rxn), the ZPE difference between species and ΔH_0(Th), the ZPE reaction threshold. The other terms shown in Figure 3 refer to values at 298 K. A PES feature is a stationary point on the PES and is either a minimum (or well), a saddle point (transition state) or an association/dissociation limit. The ZPE of each feature is stored in the property `me:ZPE`. It is important to emphasize that the relative locations are defined in terms of the ZPE 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 enthalpy of reaction at 0 K.

The ZPE are calculated from data in the input file in one of a number of conventions:

(1) In the *arbitrary* convention the zero is chosen by the user, often the ZPE of the lowest energy species. The user is responsible for ensuring that all the input data is referred to this zero and will, in effect, input values of ΔH_0 (Rxn) and ΔH_0 (Th) energy relative to the ZPE of this species.

(2) In the *computational* convention, the energies of all the species and transition states are related to the energy of the separated electrons and nuclei. This convention is often used in the output from electronic structure calculations.

(3) In the *thermodynamic* convention, the data are input as enthalpies of formation. If values at 0 K ($\Delta_f H_0$) are used, MESMER calculates ΔH_0 (Rxn) directly from the appropriate differences. If $\Delta_f H_{298.15}$ are used, MESMER converts them to the 0 K values, using input molecular properties, before calculating ΔH_0 (Rxn). Enthalpies of formation can be obtained from thermodynamic databases or from electronic structure calculations. The former do not contain data on transition states, which must be either estimated or taken from calculations, recalculating the values, if necessary, so that they are related to enthalpies of formation.

The energy convention is set by the first molecule in the list, as follows: MESMER looks for a `<property>` with a `dictRef="me:ZPE"` or `dictRef="me:Hf0"` or `dictRef="me:Hf298"` and will use the first of these found. The last two provide the energy as a Heat of Formation at 0K or at 298K and set the energy convention to thermodynamic. The meaning of a property with `dictRef="me:ZPE"` depends on its convention attribute, which can be computational or thermodynamic.

```
<property dictRef="me:ZPE">
<scalar units="Hartree" convention="computational">-37.791070</scalar>
</property>
```

If there is no convention attribute the energy convention is arbitrary.

Once the convention is set to either thermodynamic or computation, subsequent energies are automatically converted to it. If a property with `dictRef="me:Hf0"` or `dictRef="me:Hf298"` has been used to set the energy, a property with `dictRef="me:ZPE"` is written to the output XML file with an attribute describing the origin of the data. If the output XML file is used in

subsequent calculations for input, the ZPE data are used, rather than the original property, because the priority order for defining the energy is `me:ZPE`, `me:Hf0`, `me:Hf298`.

Because MESMER is based on ZPE values, care must be taken when using data obtained from *ab initio* methods. Some calculations output the classical potential energies (see ΔH_{cl} (Rxn) in Figure 3) must be adjusted for ZPE. As an alternative to doing this manually, MESMER will do it automatically if `me:ZPE` is given an attribute `zeroPointVibEnergyAdded="false"`. Also, 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`. Otherwise, zero point energy is set by the first molecule in the list, as follows: MESMER looks for a `<property>` with a `dictRef="me:ZPE"` or `dictRef="me:Hf0"` or `dictRef="me:Hf298"` and will use the first of these found. The last two provide the energy as a Heat of Formation at 0K or at 298K and set the energy convention to thermodynamic. The meaning of a property with `dictRef="me:ZPE"` depends on its convention attribute, which can be computational or thermodynamic.

```
<property dictRef="me:ZPE">
<scalar units="Hartree" convention="computational">-37.791070</scalar>
</property>
```

If there is no convention attribute the energy convention is arbitrary.

In addition there are two important points that need to be accounted for when specifying zero point energy:

(1) Different units can be specified for input of the ZPE of each reaction species and MESMER will convert all the energies to the internally used cm^{-1}. However the selected units must be the same for every species in the input XML file if it is to be displayed using Firefox (see section 7.1), although the output XML file produced by MESMER will have consistent units and will display correctly.

(2) For association (and dissociation) reactions, the required ZPE of the reactants (and products) is the sum of ZPEs of the associating (dissociated) species. For example consider the association reaction, $A + B \rightleftarrows C$, the required reactant ZPE is the sum of the ZPEs of A and B. For association reactions it is usually convenient to specify

36

the excess reactant as having a ZPE of zero and ascribe the sum of the ZPEs (i.e. of A + B in the example above) to the deficient species.

Figure 3 illustrates the different energy definitions for application in MESMER. In each case 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.
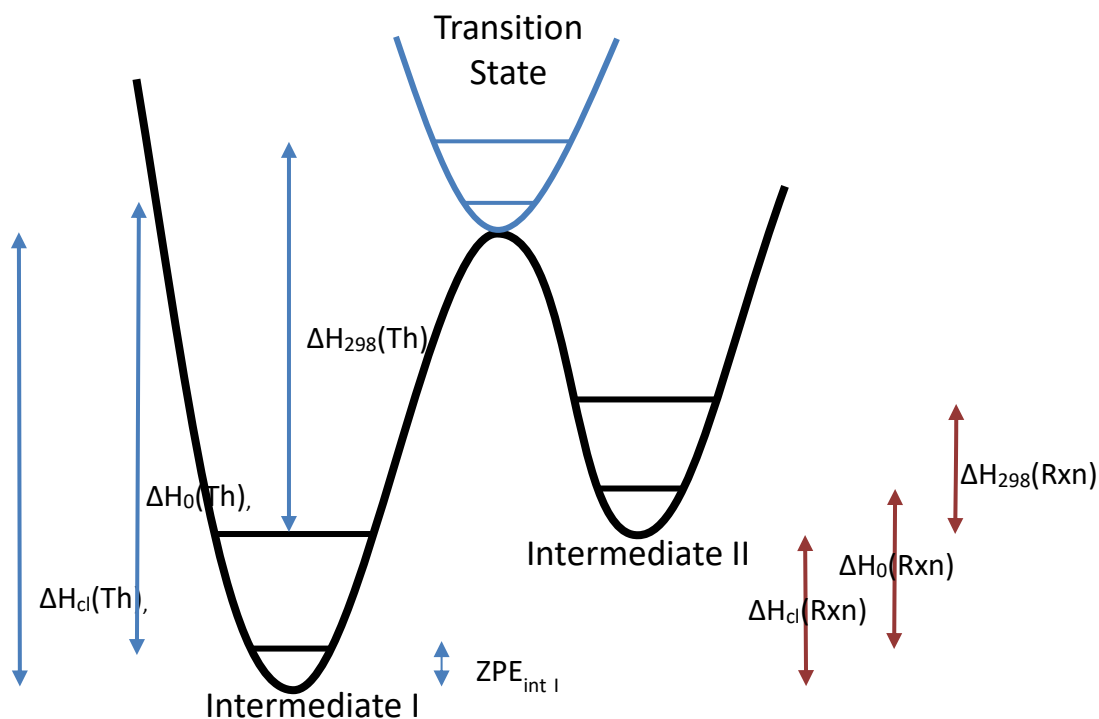
Figure 3: Schematic PES illustrating the definitions of ΔH used in MESMER

7.3.2 reactionList

`reactionList` is composed of each individual `reaction` that may occur in a ME system. Effectively, the `reactionList` specifies which `molecule`s are connected via which transition state. A `reaction` may be one of a number of types listed in Table3. The reaction type assigned for a given reaction is determined by the number of species and their species types (more detail below) involved in the reaction, and these are also listed in Table 3. If MESMER cannot assign a reaction to one these types it will stop.

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 user's choice) when the XML file is viewed in Firefox.

Reactant 1	Reactant 2	Product 1	Product 2	Reaction type
`modelled`	Absent	`modelled`	Absent	Isomerization
`deficientReactant`	`excessReactant`	`modelled`	Absent	Association
`modelled`	Absent	`sink`	`sink`	Irreversible dissociation
`deficientReactant`	`excessReactant`	`sink`	`sink`	Irreversible exchange
`modelled`	`excessReactant`	`sink`	Absent	Bimolecular sink
`modelled`	`excessReactant`	`modelled`	Absent	Pseudo-isomerization
`deficientReactant` `(=excessReactant)`	`excessReactant`	`modelled`	Absent	Second order association
`deficientReactant`	`excessReactant`	`modelled`	`excessReactant`	Bimolecular exchange

Table 3: Types of reactions.

(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 `reactant`s and one `product`, a dissociation reaction (A→B+C) has one `reactant` and two `product`s, irreversible unimolecular reactions (A→C) have one `reactant` and one or two `product`s, and irreversible exchange reactions (A+B→C+D) have two `reactant`s and two `product`s. `reaction`s may or may not have a `me:transitionState`, which is discussed further below. All `reactant`s, `product`s, and `me:transitionState`s are molecules, and have:

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

    b. A `role` (previously `me:type`), which specifies the `molecule`'s role in the ME model of the system. The possible values of `role` are given in Table 4.

role	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: Different values of role (me:type) and their corresponding definitions in MESMER

(4) If a reaction contains a species with the role of excessReactant, then the number density or concentration of this species must be given. This can be done in several places depending on the calculation being performed. For simple calculations this quantity can be declared on the reaction definition using the element me:excessReactantConc and can either be given as a concentration (in units of molecule cm^{-3}), for example:

```
<me:excessReactantConc>1.00e17</me:excessReactantConc>
```

Or as a percentage of bath gas concentration, for example:

```
<me:excessReactantConc percent="true">0.00308601</me:excessReactantConc>
```

For more complex calculations it is usually better to define this excess concentration as part of the condition statement (see section 8.3.1).

(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 [4], 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 [5] specified with WKB. More detail can be found in the section on plug-in classes.

(6) A `me:MCRCMethod` specification, which indicates how microcanonical rate constant, $k(E)$ is to be treated for a particular reaction. The details of available methods are given in section 11.2.4.

While MESMER will select the type of reactions that are appropriate to a system, it is useful to have some insight into the characteristics of these types and how they are deployed:

1. Isomerization: This reaction type describes the inter-conversion between two isomers and is used to connect two wells together. Example: Reaction R1 in …\MesmerQA\cyclopropene isomerization\Cyclopropene_isomerization_test.xml

2. Association: This reaction type is used as bimolecular source term. One of the reactants in assumed to be in excess and the deficient reactant is assumed to in a Boltzmann distribution. Because the deficient reaction is in a Boltzmann distribution is total mole fraction is represented as a single grain. More than one source term is allowed. Example: Reaction R1 in …\MesmerQA\OH acetylene association\ OH_acetylene_association_test.xml

3. Irreversible dissociation: This reaction type describes the loss from a well to an infinite sink. The products of such a reaction do not always need to be described in detail, however there are occasions, such as when using recombination ILT to calculate microcanonical rates, where this is required. Example: Reaction R1 in …\MesmerQA\i-propyl\ipropyl_test.xml

4. Irreversible exchange: This reaction type has been added to allow for the loss a bimolecular source species via a bimolecular reaction that leads to sink species. This can be important when analysing experimental data. Example: Reaction H-Abstraction in …\examples\OH_Ethene\Ethylene_abstraction.xml

5. Bimolecular sink: This reaction type applies to the situation where an isomer reacts with an excess species and is removed from the system. This differs from the irreversible dissociation above which is the spontaneous dissociation of a single species. Example: Reaction R3b in …\examples\OH-acetylene\OH_HCCH-irreversibleBim-publish.xml

6. Pseudo-isomerization: This reaction type is similar to the above but is reversible. In principle, this type can also be used as a bimolecular source term for situations where it is believed that the deficient reactant does remain in a Boltzmann distribution. The main use for this type for the case where a well species that is formed from a sequence of isomerization/source reactions reacts with another excess reactant. In implementing this type, knowledge of the fragmentation distribution of the products of this reaction is

40

required [6]. At present there two distribution functions available and these are discussed in section 11.2.7. Example: Reaction `R2` in `…\examples\Glyoxyl\Glyoxal.xml`

7. Second order association: Second order self-association reactions (such as $CH_3 + CH_3 \rightarrow C_2H_6$) cannot be linearized (as pseudo first order reactions can) and so introduce non-linear terms into the master equation, which prevent solution by usual means. However, progress can be made considering a local (in time) Taylor expansion of the non-linear master equation and examining the Jacobian associated with the linear terms. This approach is discussed in detail by Davis and Klippentstein [7], who demonstrate that time independent rate coefficients can be recovered. However, the local nature of the expansion does not permit the full evolution of the system distributions to be obtained, to obtain such distributions the equations of motion including non-linear terms must be integrated, which is not possible at present within MESMER, and so keywords that generate such distributions are blocked. Example: Reaction `R1` in `…\examples\2Methyl_to_Ethane\CH3_CH3_to_C2H6.xml`

8. Bimolecular exchange: This reaction type **approximately** models the situation where the initial unimolecular species is produced as a consequence of a bimolecular exchange reaction, for example

$$A_X + B \ \rightleftarrows C + D_X \tag{7.1}$$

where A_X and D_X are in excess, so that the system can be considered to be approximately linear with B and C as pseudo isomers. The deficient (in concentration) reactant B is assumed to be maintained in Boltzmann distribution in a similar way to that of reaction type 2. The product C is the species that can undergo further unimolecular reaction and reaction (7.1) essentially accounts for the initial distribution in which C is generated, that it is a different from of chemical activation. Knowledge of the distribution of the products as they emerge from a reactive collision is required. [6] This information can be obtained from detailed quantum mechanical scattering or classical trajectory calculations. MESMER offers approximate distributions based on the prior distribution which are the same as those offered for reactions of type 6 and are discussed in section 11.2.7. This method is unusual as a `me:MCRCMethod` is not specified, but the specification of `transitionState` a is mandatory. An example layout if this reaction type is:

```
<reaction id="R1">
  <reactant>
    <molecule ref="OH_i" role="deficientReactant"/>
```

```
        </reactant>
        <reactant>
          <molecule ref="Glyoxal" role="excessReactant"/>
        </reactant>
        <product>
          <molecule ref="HC(O)CO" role="modelled"/>
        </product>
        <product>
          <molecule ref="H2O" role="excessReactant"/>
        </product>
        <transitionState>
          <molecule ref="TS1" role="transitionState"/>
        </transitionState>
        <me:tunneling xsi:type="Eckart"/>
        <me:FragmentDist xsi:type="me:modPrior">
          <me:modPriorOrder>0.256437</me:modPriorOrder>
          <me:modPriorNexp>0.0</me:modPriorNexp>
          <me:modPriorTref>298</me:modPriorTref>
        </me:FragmentDist>
        <me:excessReactantConc>1E14</me:excessReactantConc>
        <me:excessProductConc>1E15</me:excessProductConc>
      </reaction>
```

(**Note: this reaction type is still subject to validation and should be regarded as a beta implementation.**)

7.3.3   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) One or more `me:PTs`, which is used to specifies the physical conditions at which the ME model is to be run. `me:PTs` can contain `me:PTset` elements which specify a grid of pressure-temperature like:

```
<me:PTs>
  <me:PTset units="PPCC">
    <me:Prange initial="2.42E18" final="2.82E18" increment="0.40E18"/>
    <me:Trange initial="800" final="1000" increment="200"/>
  </me:PTset>
</me:PTs>
```

where `initial` and `final` are the respective minimum and maximum value of interest for pressure or temperature, and `increment` is the interval at which values are to be written. The `factor` keyword can be used in place of `increment` to give multiplicative intervals. `factor` is useful when calculating fall-off curves, for example:

```
<me:PTs>
```

```
  <me:PTset units="PPCC " ref1="HSO2" ref2="HOSO" refReaction="R4">
   <me:Trange initial="400" final="1500" increment="100"/>
   <me:Prange initial="1.e+12" final="1.e+24" factor="10.0"/>
  </me:PTset>
</me:PTs>
```

The specification of reactants and reaction as attributes to `me:PTset` is optional but if specified will generate a table of rate coefficients for this reaction in the .log file.

Alternatively, it can contain a number of `me:PTpair` elements, which specify, as attributes, the pressure (`P`) and temperature (`T`), at which a particular master equation model is to be run.

```
  <me:PTs>
    <me:PTpair units="Torr" P="200.72" T="800." />
    <me:PTpair units="Torr" P="250.00" T="1000." />
  </me:PTs>
```

The number of `me:PTpair`s tells MESMER how many ME simulations to run. For example, three `me:PTpair`s 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 is specified in `units`. The units recognized by MESMER for pressure input are: (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`; (5) pounds per square inch, specified as `psi` and (6) moles per cm^3 as `moles/cc`. In addition to these attributes, the `precision` attribute determines the precision at which the calculation is to be run. The `bathGas` attribute specifies the bath gas. If it is omitted, the default bath gas specified in the `<me:bathGas>` element (see above) is used.

The `units`, `precision`, `bathGas` and several other attributes can be put on the `me:PTs` element to apply to all of its `me:PTpair` sub-elements. (`PTset` sub elements are not affected by these.) Note, if the attribute `bathGas` is declared in this element, the specified value will take precedence over the global value specified outside of the `me:PTs` element.

This element can also be used to specify experimental data, to be used for fitting; more detail can be found in section 8.3.

(3) `me:InitialPopulation` specifies the initial population for each species for the calculation of the time evolution of the system. If this keyword is omitted then: 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

43

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

7.3.4 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 for specification of ZPE. For a convergent solution of the ME utilizing the exponential down model, the grain size should be smaller than the $\langle \Delta E \rangle_d$, the average energy transferred in a deactivating collision, values of any of the `modelled` molecules.

(2) `me:automaticallySetMaxEne` automatically determines the upper limit to the span of energy grains. MESMER automatically increases highest energy grain in the system until the largest population in any species is smaller than the threshold defined. For example, `<me:automaticallySetMaxEne>1e-15</me:automaticallySetMaxEne >` will set the threshold population 1×10^{-15}.

(3) `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. However, 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. The `me:energyAboveTheTopHill` value can be set automatically using the `me:automaticallySetMaxEne` keyword which is described in section 7.3.5.

44

(4) `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`.

7.3.5 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.

  `me:eigenvalues`, 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 `<me:eigenvalues>10</me:eigenvalues>` is specified, then eigenvalues `ngrn` to `ngrn-10` will be printed. The specification, `<me:eigenvalues>all</me:eigenvalues>` 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. `analyticalRepresentation`: calculates a Chebyshev analytical representation of the rate coefficients over a specified temperature and pressure range.

f. `ThermodynamicTable`: calculates thermodynamic functions for the species defined in the molecular list.

g. `sensitivityAnalysis`: analyses the ME model in terms of the sensitivity of the calculated outputs in terms of the variance of a set of specified parameters.

- `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 standard 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 on a logarithmic time scale. Note this control is not supported for systems that contain a second order self association reaction term (i.e. $2A \rightarrow A_2$).

- `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, as calculated by summing over energy grains, on a logarithmic time scale. Note this control is not supported for systems that contain a second order term.

- `me:printPhenomenologicalEvolution` which prints out time evolution of every species on a logarithmic time scale. This control differs from `me:printSpeciesProfile` as the profiles are calculated using standard rate law expressions and the phenomenological rate coefficients calculated from the Bartis-Widom algorithm (see section 9.1.1.6). If the CSE and IERE eigenvalue sets are well separated the results from this method and the above should match closely.

Note this control is not supported for systems that contain a second order self association reaction term (i.e. $2A \rightarrow A_2$).

- `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. The temperature and interval of this test can be modified by specifying the attributes `Tmin` (lowest temeprature, default 200 K), `Tmax` (highest temperature, default 2000 K) and `Tstep` (the temperature interval) e.g.:

  `<me:testMicroRates Tmin ="150.0" Tmax ="1500.0" Tstep ="50.0"/>`

- `me:testRateConstants`, which computes TST $k(T)$s for each reaction in the system.

- `me:useTheSameCellNumberForAllConditions`, if set or set to true 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 and will tend to reduce the number of times densities of states are calculated.

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 Firefox diagram. There is a control on the diagram that allows this to be toggled.

- `me:automaticallySetMaxEne` which automatically increases `me:energyAboveTheTopHill` until the largest population in any species is smaller than the threshold defined. For example

  `<me:automaticallySetMaxEne>1e-15</me: automaticallySetMaxEne >`

  will set the threshold population 1×10^{-15}.

- `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.

- `me:ForceMacroDetailedBalance`. This keyword forces macroscopic detailed balance when using the Bartis-Widom method to calculate macroscopic rate coefficients. This facility has been added for those situations where numerical inaccuracy yields negative rate coefficients, imposing macroscopic allows an alternative implementation of the Bartis-Widom method to be used which is less susceptible to these difficulties. Strictly speaking, an irreversible system does observe microscopic reversibility and does not rigorously obey detailed balance, so this method should be used with care. Please note: further to user feedback and as of MESMER 5.1, this feature is imposed and the presence of absence of this keyword will have no effect. To revert to the pre-MESMER 5.1 behaviour, users should set the value of the element `me:ForceMacroDetailedBalance` to `false` in the defaults.xml file, after which this keyword will be active again. To use this keyword insert the following line into the control section:

  `<me:ForceMacroDetailedBalance />`

It is possible to use more than one `me:control` block. For example, the first control block may use a marquardt calcMethod to fit some parameters; a second may use analyticalRepresentation to output a Chebeshev representation of the rates, and a third may use ThermodynamicTable to output the thermodynamics of the molecules.

There can also be multiple me:conditions blocks. A block is used by its corresponding me:control block. If there are more control than conditions blocks, the last `me:conditions` block is reused.

7.4   Summary Table: Molecular input variables in MESMER

Table 5 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)		●	●	●	●	●
v_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						●

Table 5: Essential Variables in MESMER Calculations

8   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.

8.1   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" />
```

The first representation can be extended by inserting child elements 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.

8.2 Notes on Input File Structure

The following are a set of notes on the general structure of a MESMER input file. Note the format of the MESMER file has changed between versions 2.0 and 3.0 in order that an XML schema can be written. However, MESMER 3.0 should be compatible with earlier version of the input files.

- Use lower camel case for new element and attribute names (except for obvious abbreviations like the attributes P and T). This helps to make the capitalisation more predictable.

- Attribute names are generally unqualified, i.e. the namespace "`me:`" should not be used. Using this style fits better with CML and common XML practice. E.g.: an experimental pair should be expressed as `<me:PTpair units="PPCC" P="30.0e16" T="720.">` rather than `<me:PTpair me:units="PPCC" me:P="30.0e16" me:T="720.">` However, it has been necessary to introduce a qualified attribute name to handle plugins with data (see below), as in `xsi:type="me:HinderedRotorQM1D"`.

- `<title>` `<description>` and `<ref>` used in Mesmer are not CML elements (just CML attributes) and so `<me:title>` `<me:description>` and `<me:ref>` should be used instead.

- Because it is not possible to add validatable extra structure to CML elements it is necessary to use `<me:molecule>` rather than `<molecule>` in

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

- The `<mesmer>` element should have an extra attribute `xmlns:xsi= http://www.w3.org/2001/XMLSchema-instance`. This is used as an extensible way of validating the elements of those plugins which have their own data.

- Plugin classes (see section 11.2) with data should have the data as sub-elements of the plugin type. So

```
<me:MCRCMethod>MesmerILT</me:MCRCMethod>

<me:preExponential  lower="0.9e-11"  upper="9e-11"  stepsize="1e-
12">6.78e-11</me:preExponential>

<me:activationEnergy>0.0001</me:activationEnergy>
```

becomes

```
<me:MCRCMethod name="MesmerILT" xsi:type="MesmerILT">

 <me:preExponential  lower="0.9e-11"  upper="9e-11"  stepsize="1e-
12">6.78e-11</me:preExponential>

 <me:activationEnergy>0.0001</me:activationEnergy>

</me:MCRCMethod>
```

Plugins without data can continue to use the old format:
```
<me:MCRCMethod>RRKM</me:MCRCMethod>
```

but writing a schema that validates this has not proved possible. So, use the new form even without data:

```
<me:MCRCMethod name="RRKM"/>
```

 Mesmer can still parse the old form, with or without data.

- The molecule names in MESMER (attribute id in <molecule> and ref elsewhere) do not always conform to the requirement in CML which is that they should start with a letter or an underscore, contain only ASCII letter and numbers plus '-' and '_' i.e. with no other punctuation or whitespace. In addition, they can have a prefix (like opt1:TS2). This has been relaxed for Mesmer but it would be good practice to make new names conform.

Note also that the attribute name `me:type` is replaced in MESMER 3.0 by `role` (which already appears in CML). So

```
<molecule ref="O2" me:type="excessReactant" />
```

becomes

```
<molecule ref="O2" role="excessReactant" />
```

Finally, the CML schema CMLschema3.xsd has been modified in a few places, making it slightly less restrictive. Search for "MESMER" in CMLforMESMER.xsd.

54

8.3 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 these data is as child elements of the `me:PTpair` element described in section 7.3.3, 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 of χ^2 defined by:

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{\left(k_{i,ob}(p_i, T_i) - k_{i,mod}(p_i, T_i)\right)^2}{\sigma_i^2} \right) \tag{8.1}$$

where $k_{i,ob}(p_i, T_i)$ is the experimentally observed value, $k_{i,mod}(p_i, T_i)$ is the modelled value for the same conditions and σ_i is an estimate of the error in the observed value. χ^2 is minimized to determine the best fit model parameters. The details of specification of the experimental conditions and observed measured measurement for each type of experimental value are discussed in this section.

8.3.1 Specification of Biomolecular Source Terms

The general approach adopted in MESMER for the description of bimolecular source terms follows that used in a number of experiments analyses, which is to assume that one of the reactants is in excess and so the ME can be linearized (see section 14.1.1). To use this approach, it is necessary to specify the concentration of the excess species. This can be done globally as part of the reaction definition (see section 7.3.2). However, when analysing experimental data it is often the case that the concentration of the excess species is specifically for each set of conditions and MESMER allows the concentration of the excess species to be defined as part of a `me:PTpair` definition using the attribute `excessReactantConc` for example:

```
<me:PTpair units="torr" P="114.2" T="723 " excessReactantConc="5.01E+15"
precision="d" bathGas="N2"> …
```

The excess species concentration has units of molecules/cm^3 and, if specified, the value of this attribute will take precedence over the value specified in the reaction definition. Alternatively, the excess species concentration can be specified as a percentage of the bath gas concentration

by including the attribute `percentExcessReactantConc` which causes the value of the attribute `excessReactantConc` to be interpreted as a percentage, for example,

```
<me:PTpair units="torr" P="114.2" T="723 " percentExcessReactantConc="true"
excessReactantConc="0.01" precision="d" bathGas="N2"> …
```

Some more complex systems have more than one excess species in this case the excess concentration can be specified using an array, for example,

```
<me:PTpair units="torr" P="1000" T="298.0">
  <me:excessReactantConcArray>
    <me:excessReactantConc reactionRef="R1" concentration="0.00308601"
percent="true"/>
    …
  </me:excessReactantConcArray>
</me:PTpair>
```

8.3.2 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 the experimentally measured rate coefficient for acetyl (CH_3CO) loss is compared to the phenomenological rate coefficient calculated by MESMER:

```
<me:PTpair units="Torr" P="200.12" T="500" 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. In addition to the references, there is an `error` attribute which must be present when experimental data are added. If all `error` values are greater than zero for a given fitting exercise then, it is assumed that they are independent error estimates and MESMER will use these errors to estimate a goodness-of-fit indicator based on the χ^2 distribution. If one or more `error`s are less than or equal to zero, MESMER will assume that there is no independent error estimates and will attempt to estimate errors of the fitted parameters by adjusting their values so that $\chi^2/N = 1$ where N is the number of degrees of freedom, though, obviously, with this latter approach there is no independent check on the quality of the model.

For bimolecular reactions, it is necessary to specify either the concentration of the excess species or the initial concentration of the associating species in the reaction definition. To obtain a bimolecular rate coefficient this concentration needs to be accounted for and this is done by using the keyword `refReaction` which associates the experimental data with a reaction definition that specifies this concentration. The following example is taken from the fitting of experimental data for the reaction $CH_3 + H \rightarrow CH_4$:

```
<me:PTpair units="Torr" P="153.5" T="504." precision="dd">
  <me:experimentalRate ref1="Methyl" ref2="Methane" refReaction="R1" error="4.68e-12">4.68e-11
  </me:experimentalRate>
</me:PTpair>
```

If there are no experimental rates to compare with, the child element can be left undefined:

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

or even shorter

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

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

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

The keyword `refReaction` is also used for the situation where the reaction of interest is a sink reaction whose unimolecular reactant can decay through more than one channel. For example, the $CH_3 + OH$ reaction proceeds via CH_3OH but there are several exit channels for the initially formed excited species one of which is $CH_3OH \rightarrow CH_2 + H_2O$. If this reaction has the name "R2" the rate coefficient might be expressed as:

```
<me:experimentalRate ref1="CH3OH" refReaction="R2" error="6.e9">6.42e10</me:experimentalRate>
```

8.3.3 Experimental Yields

Experimental yields can be specified in the following way:

```
<me:PTpair units="PPCC" P="4.46E18" T="195" 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.

8.3.4 Experimental Eigenvalues

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

```
<me:PTpair units="PPCC" P="7.6E18" T="440." 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.

8.3.5 Trace analysis

MESMER can calculate the evolution of species as function of time and this feature can be used to analyse experimental traces (time series) directly. Trace data is input as follows:

```
<me:PTpair units="torr" P="114.2" T="7.23E+02 " startTime="0" excessReactantConc
="6.64E+15" timeUnits="millisec" precision="d" default="true" bathGas="N2">
  <me:rawData ref="OH" diffusiveLoss="80" name="Date: 05/07/2017 Expt No.: 1 Trac
No.: 1">
    <me:times>0.006 0.012 0.018 0.024 0.03 0.036 0.042 0.048 0.054 0.06 0.066 …
1.152 1.158 1.164 1.17 1.176 1.182 1.188 1.194 1.2</me:times>
    <me:signals>0.001906 0.001510 0.001569 0.001449 0.001254 0.001311 0.001461 …
0.000032 0.000021 0.000101 0.000141</me:signals>
  </me:rawData>
</me:PTpair>
```

The definition of the time series data begins, as with other experimental data, with the definition of the conditions at which the experiment was conducted but includes the additional term `startTime` which indicates the time reaction was initiated. The child element `me:rawData` has several important attributes: `ref` defines the species that is being observed in the experiment, in this case OH; `diffusiveLoss` is the coefficient of loss of the observed species through means other than direct reaction, typically diffusion from the observation region, this term adds a linear loss term to the transition probability matrix for the species under observation, the units of this term are s^{-1}; `name` is the used to identify the trace being analysed. The element `me:rawData` has two child elements: `me:times`, this is array of times at which observations of the species were made and `me:signals`, an array of signals of the observed species.

Alternatively, if diffusive loss data are known for more than one species in the system these can be defined in a diffusive loss array, for example,

```
<me:PTpair units="torr" P="114.2" T="7.23E+02 " startTime="0" excessReactantConc
="6.64E+15" timeUnits="millisec" precision="d" default="true" bathGas="N2">
  <me:rawData ref="OH" name="Date: 05/07/2017 Expt No.: 1 Trac No.: 1">
    <me:diffusiveLossArray>
      <me:diffusiveLoss speciesRef="C2H4" loss="12.5"/>
      <me:diffusiveLoss speciesRef="OH" loss="25.1"/>
    </me:diffusiveLossArray>
    <me:times>…</me:times>
    <me:signals>…</me:signals>
  </me:rawData>
</me:PTpair>
```

The difference here is that a single loss term has been replaced by an array of loss terms, `<me:diffusiveLossArray>`. The loss terms are assumed to have units of s^{-1}.

When fitting trace data, MESMER will first normalize the calculated trace, so that the signal amplitude best fits that of the observed date, using a simple linear regression. Then the temporal dependence of the trace is then optimized by adjusting fitting parameters in the usual way using, for example, the Marquardt algorithm.

Typically, experimental traces come with fluctuations, which can arise from several sources. It is, therefore, often desirable to weight traces to account for these fluctuations in such a way that those traces that exhibit most noise make less of a contribution to the χ^2 merit function. A trace weight can be introduced by specifying the attribute `weight`, for example:

```
<me:rawData ref="OH" diffusiveLoss="80" name="Date: 05/07/2017 Expt No.: 1 Trac
No.: 1" weight="4.1829e-09">
```

An obvious question is how are such weights obtained? If `weight` is not specified, it is assumed to be unity and is such a case MESMER will generate weights as part of a fitting calculation. The weight of trace is calculated as the average of the sum of square residuals between the experimental and calculated trace points

$$s_i = \frac{1}{N_i - 1} \sum_{j=1}^{N_i} \left(f_{e,ij} - f_{c,ij} \right)^2 \tag{8.2}$$

where s_i is the reciprocal weight of trace ith trace which has N_i points, $f_{e,ij}$ is the value of the jth point of the ith experimental trace and $f_{c,ij}$ is the corresponding value for the calculated trace. The s_i are written to the `weight` attribute in the output file. To apply these weights a second fitting calculation must be run this time with the keyword `me:useTraceWeighting` set in the control section, for example:

```
<me:control>
  <me:calcMethod xsi:type="me:marquardt">
    <me:MarquardtIterations>20</me:MarquardtIterations>
    <me:MarquardtTolerance>1e-6</me:MarquardtTolerance>
    <me:MarquardtDerivDelta>0.025</me:MarquardtDerivDelta>
  </me:calcMethod>
  <me:useTraceWeighting />
</me:control>
```

This will cause the weights to be calculated via the normalization

$$w_i = \frac{1/s_i}{(1/N_T)\sum_i 1/s_i} \tag{8.3}$$

where is the number of traces, so that the noisier the data the less its weight will be. These weights are then applied to the trace contribution to the overall merit function χ^2. An example of trace fitting can be found in the examples folder under OH_Ethane.

It is important to note that, once the fitting is complete any further analysis, for example the generation and analytical representation of rate coefficients, should be done without the diffusion loss terms included, that is they should be done in a separate calculation.

8.4 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:

```
precision="quad-double"
```

```
precision="double-double"
```

or

```
precision="qd"
```

```
precision="dd"
```

8.5 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 the 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^{∞} (units of cm^3 molecule^{-1} s^{-1} in this bimolecular example) 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 6.201×10^{-12}), and step size (with value 2.0×10^{-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 Table 6:

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

	where T_{ref} is 298.0 K by default but can be set via the attribute `referenceTemperature`.	
v_i	Imaginary frequency used to calculate tunnelling coefficients.	`me:imFreqs`
E_x	Prior distribution energy	`me:EnergyExcess`
n_p	Prior distribution order	`me:modPriorOrder`
k_s	Simple bimolecular sink reaction	`me:BimolecularLossRateCoefficient`

Table 6: Possible fitting parameters

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 declaring one parameter to be derived from another. For example, the $\langle \Delta E \rangle_d$ parameters for two isomers can be coupled via the sequence:

```
<molecule id="Isomer1">
  .
  .
  <me:energyTransferModel xsi:type="me:ExponentialDown">
    <me:deltaEDown units="cm-1" lower="100" upper="400"
stepsize="10">174</me:deltaEDown>
  </me:energyTransferModel>
  .
  .
</molecule>
<molecule id="Isomer2">
  .
  .
  <me:energyTransferModel xsi:type="me:ExponentialDown">
    <me:deltaEDown units="cm-1"
derivedFrom="Isomer1:deltaEDown">174</me:deltaEDown>
  </me:energyTransferModel>
  .
  .
</molecule>
```

In the first part of this sequence a $\langle \Delta E \rangle_d$ parameter is defined in the usual way. In the second part of the sequence another $\langle \Delta E \rangle_d$ parameter is defined, but this time without any range attributes. Instead, the attributed `derivedFrom` indicates that this parameter is derived from

another parameter and the value of the attribute is the identity of the other parameter. The identity of the independent parameter has two parts, the name of the species (`Isomer1` in the above example) and the name of the parameter of that species (`deltaEDown` in the above example) and will be one of the XML element names in the above table, but with the `me:` prefix omitted. A linear relationship between parameters can be specified, for example,

```
.
.
<me:deltaEDown derivedFrom="Isomer1:deltaEDown"factor="1.5" addand="1.0">4</me:deltaEDown>
.
.
```

In this case the derived parameter is constrained to the value of the independent parameter by the expression:

```
Isomer2:deltaEDown = factor * Isomer1:deltaEDown + addand
```

The default value for `factor` is 1.0 and for `addand` is 0.0, and there is no limit to the number of constraints that can be applied.

9   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.

9.1   MESMER output files

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

9.1.1 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.

9.1.1.1 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}.

9.1.1.2 $k(E)$s & Tunnelling Corrections

k_f(e) grains for <reaction name>

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

<code>k_b(e) grains for <reaction name></code>

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

<code>Tunnelling coefficients for: <reaction name></code>

<code>V0 = 11201.4, V1 = 10047.3, barrier0 = 9864, barrier1 = 8834, imFreq = 5.0605e+13</code>

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.

9.1.1.3 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:

$$-K_1 A + B = 0$$

$$-K_2 B + C = 0 \qquad (9.1)$$

$$A + B + C = 1$$

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

$$\begin{pmatrix} -K_1 & 1 & 0 \\ 0 & -K_2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{9.2}$$

The equilibrium fraction of each isomer or pseudo isomer 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.

9.1.1.4 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:

$$\frac{d}{dt}\boldsymbol{p} = \boldsymbol{M}\boldsymbol{p} \tag{9.3}$$

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:

$$\boldsymbol{p}(t) = \boldsymbol{U}e^{\Lambda t}\boldsymbol{U}^{-1}\boldsymbol{p}(0) \tag{9.4}$$

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 Λ 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.

9.1.1.5 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:

$$P(t) = 1 - \sum_i \sum_E \boldsymbol{p}_i(E, t) \qquad (9.5)$$

where $\boldsymbol{p}_i(E, t)$ is the time dependent population of the energy grains spanning the entire state space of the ith well obtained from (Eq. 9.4). If only one product channel is available, then (Eq. 9.5) 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:

$$P_{ni}(t) = \int_0^t \sum_{E \in i} k_{ni}(E) \boldsymbol{p_i}(E, t) = \sum_{E \in i} k_{ni}(E)[\boldsymbol{U\Lambda^{-1}}(\boldsymbol{1} - e^{\boldsymbol{\Lambda} t})\boldsymbol{U^{-1}p}(0)] \qquad (9.6)$$

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

9.1.1.6 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 Eq. 9.3. The mathematical development of the Bartis -Widom technique implemented in MESMER is described by Robertson *et al.*[8], 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 by a set of coupled differential equations similar in form to Eq. 9.3 but considerably fewer in number, with the primary difference being the absence of a description for collisional relaxation kinetics – i.e., all the rate coefficients correspond to inter-conversion 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 $n \times n$ rate coefficient matrix $\boldsymbol{K}$ representing n coupled first order differential equations, where the matrix element k_{ab} is the rate coefficient $k_{b \to a}(T,P)$. 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 $\boldsymbol{M}$ (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 $\boldsymbol{K}$, then the phenomenological rate matrix $\boldsymbol{K}$ 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.*[8]:

`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 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 K_r matrix with labels describing the interconversion to which they correspond.

`Kp matrix:`

The 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 K_p matrix with labels describing the interconversion to which they correspond.

9.1.2 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.

9.1.3 XML output

The XML output contains a copy of the input XML file together with data specified in a control block. For example an XML output file might contain:

- 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 spreadsheets, etc. is available. This can also be produced by applying the file punch.xsl to the output XML file, see section 9.1.9.

- calculated species/time profiles (provided the `me:printSpeciesProfile` and `me:printGrainedSpeciesProfile` keywords have been specified). These are also presented graphically (described below).

- metadata, including name of user etc.

- thermodynamic tables

- sensitivity analysis data

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 the MESMER distribution contains [XSLT](#) support to allow many features of the XML output files to be presented in a more friendly way when the XML file is opened in a browser. Currently, it is necessary to use Firefox 3., or later. To see the data represented graphically (see section 7.1). As well as a schematic of the stationary points on the potential energy surface (see Fig. 2), plots of species profiles, grain distribution and mean energies as a function time can be viewed provided the keywords `me:printSpeciesProfile` and `me:printGrainedSpeciesProfile` have been specified. The Actyl + O_2 association example in the MesmerQA folder generates output that demonstrates this feature. Example plots obtained for this system are shown in Figs. 4-6.

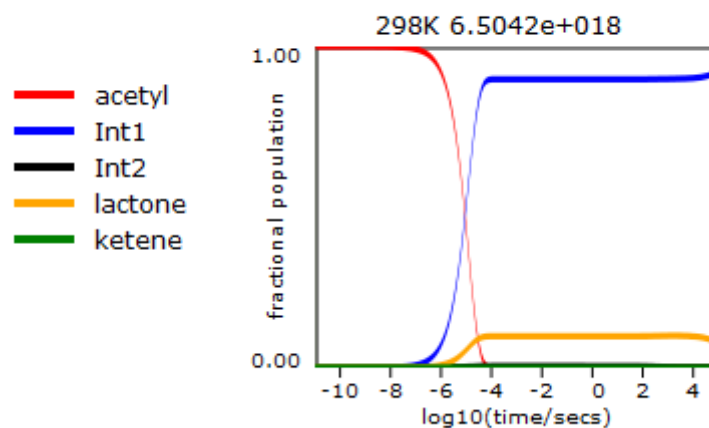**Species Populations**

Figure 4. A plot of the species in the Actyl+O2 system as a function of time. Conditions are shown at the top of the plot.
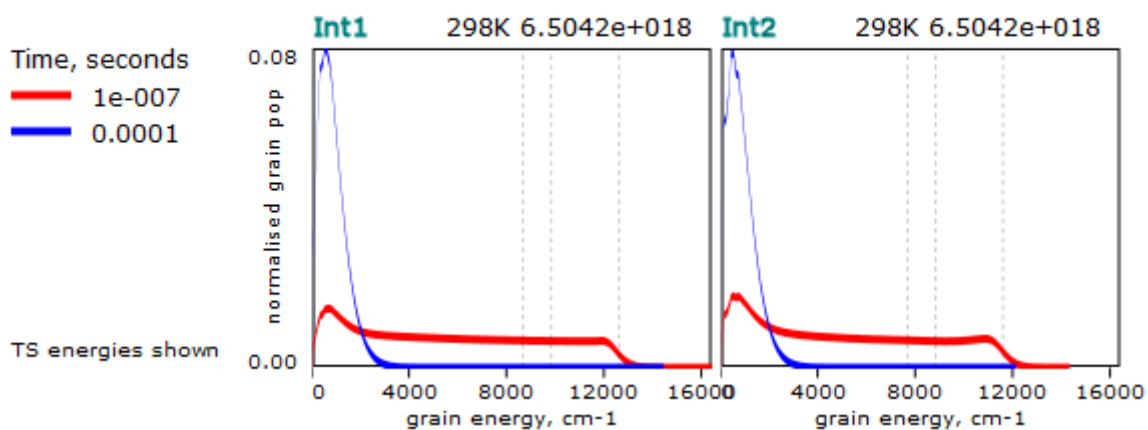
Grain Populations

Figure 5. A plot of the grain distributions of intermediate species Int1 and Int2 of the Actyl+O2 system as a function of time. Conditions are shown at the top of each plot. The dash lines correspond to reaction thresholds.

Isomer Energies

Figure 6. A plot of the mean energies species of Int1 and Int2 of the Actyl+O2 system as a function of time. Conditions are shown at the top of the plot. Other files

9.1.4 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.

9.1.5 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 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.

9.1.6 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.

9.1.7 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/).

9.1.8 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' media='screen'?>
```

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

9.1.9 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. Firefox has a facility to display the data in this form, so when Bartis-Widom output is viewed in Firefox it can be copied and pasted to other applications.

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.

10 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.

In addition there is a folder called `examples` which is contains a number of examples. 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.

10.1 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!

10.1.1 1-Pentyl Isomerization

Figure 7: PES schematic for 1-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 to 1-pentyl in an argon bath gas using a standard RRKM treatment at the transition state. Further information on this system can be found in Robertson *et al.*[8]

10.1.2 Cyclopropene Isomerization + Reservoir State
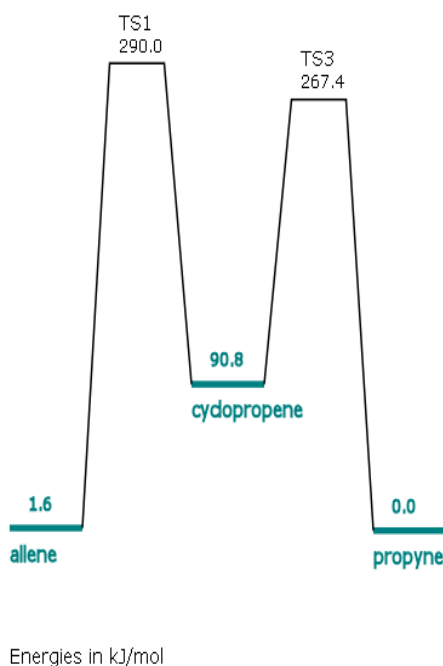
Figure 8: schematic of the PES used for the cyclopropene isomerization test

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 similar to the above system however; it includes reservoir states for each isomer. Further information on reservoir states can be found in section 4.4.2 of Glowacki *et al.*[1] and in section 14.2.1 of this manual.

10.1.3 H + SO₂
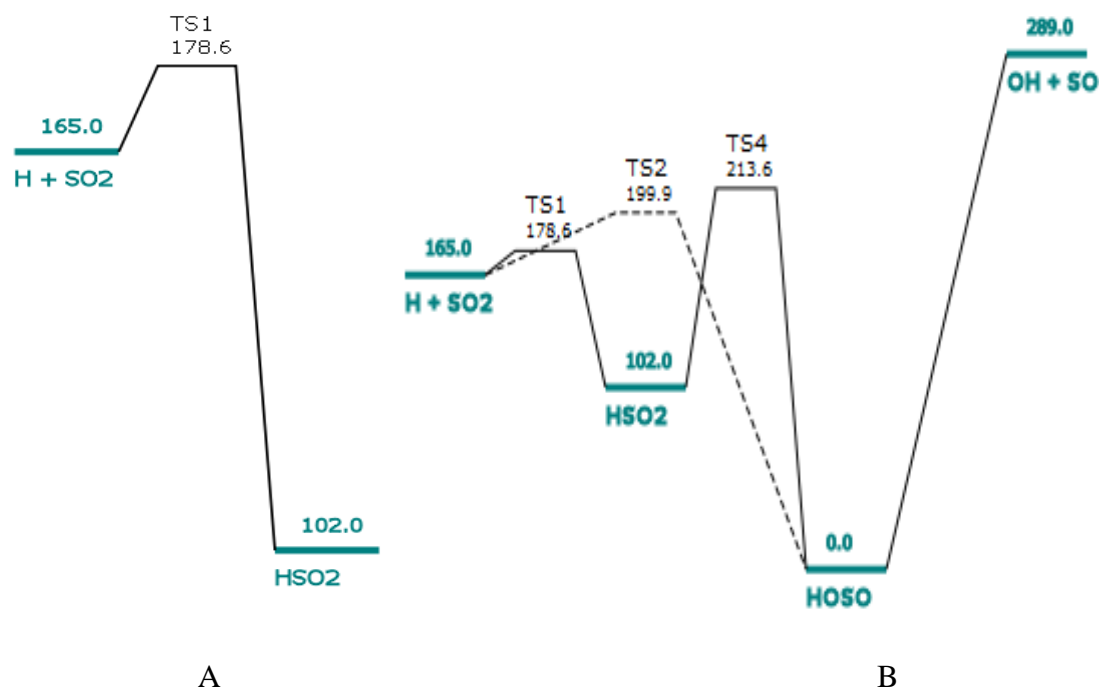
Figure 9: PES schematics for H + SO₂ test system

The folder 'HSO2/' contains two input files, `HSO2_test.xml` (A above) and `HSO2.xml` (B above). The `HSO2_test.xml` 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. In addition this example contains repeated calculations at increasing precision. The `HSO2.xml` system is an extension of the first, including more wells and demonstrating a source term contributing to multiple wells.

10.1.4 OH + C$_2$H$_2$
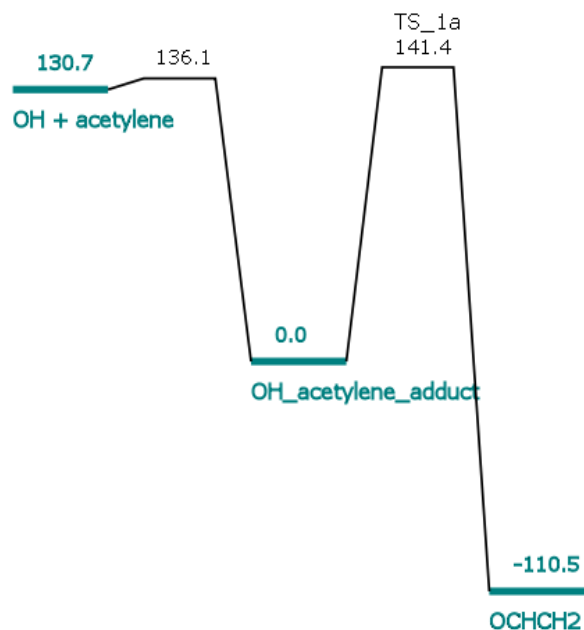
Figure 10: PES for OH + C$_2$H$_2$ test job

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.

10.1.5 CH₃CO + O₂

Figure 11: PES for CH₃CO + O₂

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.

10.1.6 2-propyl (*i*-propyl)
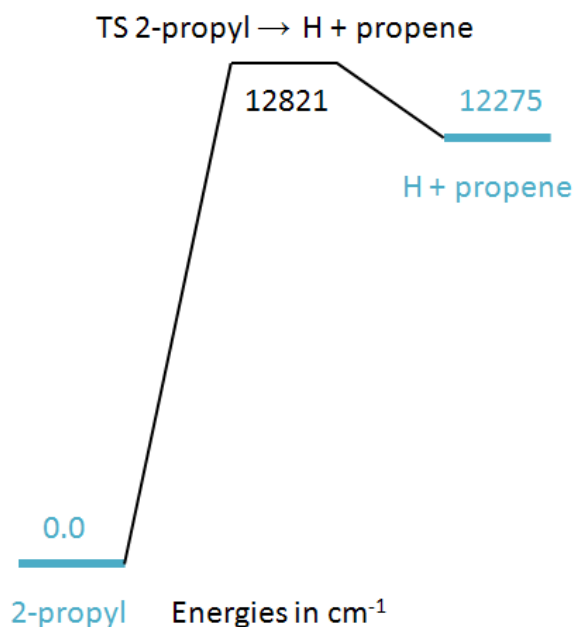
Figure 12: PES for 2-propyl (*i*-propyl) dissociation

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.

10.1.7 Benzene-OH Oxidation
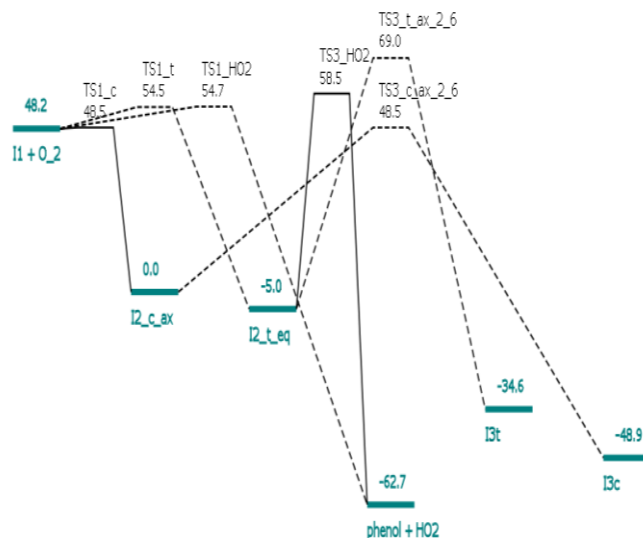
Figure 13: PES for benzene-OH + O_2

The `benzene_oxidation_test.xml` input file is located within the folder 'benzene_oxidation/'. This system includes a bimolecular source (I1 + O_2), 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 $\rightarrow$ phenol + HO_2), and one is an irreversible exchange reaction (benzene-OH + O_2 $\rightarrow$ phenol + HO_2). The $k(E)$s for each channel are calculated using RRKM theory.

10.1.8 Thermodynamic Table

The `ThermodynamicTable.xml` input file in the folder ThermodynamicTable generates thermodynamics function data 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.

10.1.9 UnitTests

This is a set of tests that are used by developers during MESMER development to test some lower level aspects of MESMER.

81

10.2 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.

Example	Description	Significant Keywords
2Methyl_to_Ethane	$2CH_3 \rightleftarrows C_2H_6$: demonstrates the simulation of a second order reversible reaction. Rate coefficients derived from the Jacobian obtained by expansion of the non-linear master equation. A short fitting exercise is performed. (See ref. Blitz et al.[9])	`me:marquardt`
AcetylO2	An extended example of the Actyl + O_2 system presented in the MesmerQA	`me:RRKM`
AcetylPrior	$CH_3CO + O_2 \rightarrow CH_3C(O)O_2$ (Sink) $CH_3CO \rightarrow CH_{3+}CO$: This example demonstrates the bimolecular sink reaction type.	`SimpleBimolecularSink` `me:InitialPopulation`
AnalyticalRepresentation	$CH_3OCH_2 + O_2 \rightarrow$ products: This contains examples of the analytic representation feature, for both Chebyshev and Plog representations.	`me:analyticalRepresentation`

Butan-1-ol	$C_4H_9OH \rightarrow C_4H_9O + H$; $C_4H_9OOH \rightarrow C_4H_9OO + H$: These examples demonstrates the automated calculation of Lennard-Jones parameters and the BiExponentialDown model.	`me:BiExponentialDown`
		 `me:MesmerILT`
		 `me:ThermodynamicTable`

Butyl_H_to_Butane	$C_4H_9 + H \rightarrow C_4H_{10}$: demonstrates the projection of internal rotor modes from a Hessian and application of hindered rotors.	`me:hessian`
		 `me:HinderedRotorQM1D`
		 `me:MesmerILT`
		 `me:ThermodynamicTable`

C4H9O2_NO2_to_C4H9O2NO2	$C_4H_9O_2 + NO_2 \rightleftarrows C_4H_9O_2NO_2$: This example demonstrates the projection of internal rotor modes from a Hessian, the fitting of experimental data and the generation of an analytical representation. (See ref. McKee et al. [10])	`me:hessian`
		 `me:marquardt`
		 `me:analyticalRepresentation`

CH3O2_NO2_to_CH3O2NO2	CH_3O_2 + NO_2 $\rightleftarrows$ $CH_3O_2NO_2$: Similar to the above example but a thermodynamic table is generated in place of the analytical representation. (See ref. McKee et al. [10])	me:hessian me:marquardt me:ThermodynamicTable
cis_to_trans_But-2-ene	The system modelled is these examples is the cis to trans isomerization of but-2-ene, with an additional loss channel from cis-but-2-ene to butadiene and H_2. It is based on the model described by W. Tsang et al. [11] In the example cis_to_trans_But-2-ene.xml derived macroscopic constants are used to determine the macroscopic evolution of the system, which can be compared with the evolution of the system as calculated from the ME. The example input file cis_to_trans_But-2-eneEx.xml is similar but uses an excited initial distribution.	me:printPhenomenologicalEvolution me:Excitation
CoupledRotors	This folder contains a number of examples of both coupled and uncoupled hindered rotor systems for various alkane species. Files marked _CCR contain examples where the full internal and external coupling rotational is treated in the classical limit. Files marked _UQR contain examples of decoupled internal rotors treated quantum mechanically.	me:ClassicalCoupledRotors
DefinedTunellingCoefficients	CH_3OH + OH $\rightarrow$ CH_3O...H_2O: demonstrates the use of defined tunneling coefficients.	me:DefinedTunnelingCoefficients
diamond	Chemi-desorption of a Methyl group from a diamond surface. This investigative example models the change in energy of a region around the attached methyl, the rate of desorption being dependent on this energy. To do this is uses an alternative energy transfer kernel based on a Gaussian function [12] .	<me:energyTransferModel xsi:type= "me:gaussian">
ErrorPropagation	i-C_3H_7 $\rightarrow$ H + C_3H_6: This example demonstrates the use of multiple control and condition blocks. A Levenberg-Marquardt fit is done followed by a prediction and error estimation on some conditions that lie outside of the experimental range.	me:ErrorPropagation

83

Ethyl_H_to_Ethane	$C_2H_5 + H \rightarrow C_2H_6$: demonstrates the application of hindered rotors using an ab initio potential.	`me:HinderedRotorQM1D`
Glyoxal	The reactions of CHOCO, as produced by the reaction CHOCHO + OH → CHOCO + H_2O, with O_2: this example demonstrates the reversible pseudo-isomerization reaction. This example does take a long time to run.	`me:FragmentDist` `me:modPrior`
H2Ominimal	H + OH → H_2O: This simple association demonstrates the use of the librarymols.xml file.	`me:MesmerILT`

i-propyl	i-C_3H_7 → H + C_3H_6: These examples extend the MesmerQA job of the same name so as to perform a fitting of the experimental data obtained by Seakins et al.[13] They demonstrate the use of an analytic hindering potential.	`me:ExponentialDown`
		`me:HinderedRotorQM1D`
		`fitting` (ipropyl_test.xml)
		`marquardt` (ipropyl_LM.xml)

Methyl_H_to_Methane	The two examples presented here are both based on the association reaction $$CH_3 + H \rightarrow CH_4$$ They both fit the data obtained by Brouard et al.[14] The file Methyl_H_to _Methane.xml gives an example where the fitted variables are the Arrhenius parameters for the association reaction. Methyl_H_to_Methane_FTST.xml gives an example where pre-computed transition state sums states obtained by Aubanel and Wardlaw [15] are used and the fitted parameters are the reaction threshold and the $\langle \Delta E \rangle_d$ parameter.	`me:DefinedSumOfStates`
OH_Ethene	This example fits master equation decay curves to experimental trace data.	`me:rawData` `diffusiveLoss`
OH_NO_to HONO	This example fits experimental data using the Marquardt algorithm.	`me:marquardt` `me:MesmerILT`
OH-acetylene	OH + C_2H_2 → CHCHOH + O_2→ products: This is an example of a reaction containing a bimolecular sink term.	`me:BimolecularLossRate-Coefficient`
POandO2	PO + O2 → OPO +O and other products: Deep well system at low temperatures. (This example was	`precision="qd"`

	developed by J.M.C. Plane and co-workers)	
reservoirSink	Actyl + O_2 system using a reservoir state.	`me:reservoirSize`

SensitivityAnalysis	There are two examples: the pentyl_isomerization_SA.xml example is a based on the isomerization of 1-pentyl to 2-pentyl and returns sensitivity indices and related data for a uniform distribution. The ipropyl_SA.xml example is based on the dissociation of the i-propyl radical and returns indicies based on a correlated distribution.	`me:sensitivityAnalysisSamples`
		`me:sensitivityAnalysisOrder`
		`me:sensitivityNumVarRedIters`
		`me:sensitivityVarRedMethod`
		`me:sensitivityCorrelatedData`

spin_forbidden_kinetics	Examples showing how to invoke non-adiabatic variants of RRKM theory for treating both intersystem crossing and internal conversion. This folder includes three different input files: one invokes Landau Zener corrections (LZ_test.xml) to treat intersystem crossing kinetics; one invokes LZ theory along with WKB corrections to treat tunnelling dynamics that occur during inter-system crossing (WKB_test.xml) [16], and the last (ZN_test.xml) invokes Zhu-Nakamura theory to treat internal conversion.[17-18] The modelled systems are simple isomerizations of singlet cyclopentyne to the lowest lying triplet via the DFT calculated minimum energy crossing point (MECP), and internal conversion of HPALD to give an OH dissociation product.	`ZhuNakamuraCrossing`
		`LandauZenerCrossing`
		`WKBCrossing`

Tunnelling	A model tunnelling system based loosely on the $H + H_2 \rightarrow H_2 + H$ and using the WKB tunnelling method.	`me:tunneling`

11 Adding Functionality to MESMER

MESMER has been written in a manner that anticipates future developments, so that as the functionality of MESMER increases, it is expected that this portion of the manual will 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 have 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 classes 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 inverse Laplace transform method for calculating microcanonical rate coefficients has been extensively commented as a guide. The code is located in the files:

- `...\src\MicroRate.h`: this file contains the definition of the abstract base class for microcanonical rate coefficient models, `MicroRateCalculator`.

- `...\src\plugins\MesmerILT.cpp`: contains the definition and the implementation of the `MesmerILT` class, which handles inverse Laplace transform calculations.

11.1 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 are 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:

11.1.1 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.

11.1.2 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.

11.1.3 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 `false` and its use makes the code more understandable.

11.1.4 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.

11.1.5 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.

11.1.6 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.

11.2 Plug-in Classes

11.2.1 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, this method 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, the data generated are plotted as a contour map to help locate 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 estimate 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 of 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. These data are written to the log file.

`analyticalRepresentation`: This class calculates an analytic representation of the rate coefficients. Two types of representation are possible: one based on Chebyshev polynomials and on based on the Plog method. The specification of which representation is required is done through the `me:format` element, which should be the first element following the

`me:calcMethod`. The default method is Chebyshev. It should be noted that if multiple reactions are defined by identical reactants and products but different transition states, the analytical representation code will only produce Cheybeshev Polynomials for one of these reactions. As such, in the rare case that multiple, distinct transitions states link the same reactants and products, different product names should be used in order to differentiate the channels (e.g prod1, prod2, …). Chebyshev polynomials are produced, for a specified ranges of temperature and pressure [19]. The aim of this method is produce text that can be used directly, or with only minor modification, in macroscopic simulation using existing chemical kinetics packages. The text is written to the .log file. The input to this method is:

```
<me:calcMethod xsi:type="me:analyticalRepresentation">
  <me:format rateUnits="cm3mole-1s-1">cantera</me:format>
  <me:precision>dd</me:precision>
  <me:chebNumTemp>7</me:chebNumTemp>
  <me:chebNumConc>5</me:chebNumConc>
  <me:chebMaxTemp>1000</me:chebMaxTemp>
  <me:chebMinTemp>200</me:chebMinTemp>
  <me:chebMaxConc units="atm">10.</me:chebMaxConc>
  <me:chebMinConc>0.001</me:chebMinConc>
  <me:chebTExSize>6</me:chebTExSize>
  <me:chebPExSize>4</me:chebPExSize>
</me:calcMethod>
```

The element `<me:format>` specifies the format which the Chebyshev coefficients should be written out in. There are at present two supported formats "`cantera`" and "`chemkin`" which refer respectively to the Cantera suite of simulation tools (see http://cantera.github.io/docs/sphinx/html/index.html) and the Chemkin application (see http://www.reactiondesign.com/products/chemkin/chemkin-2/). The attribute `rateUnits` allows the units of second order reactions to be specified, the present available choices being `cm3mole-1s-1` or `cm3molecule-1s-1`, the former being a suitable choice for both Chemkin and Cantera. The element `me:precision`, common to both representations, specifies the precision to be used for the calculation of rate coefficients that will be the basis of the analytical representation. The allowed values are the same as those described in section 8.4. Details of the other Chebyshev child elements are given in Table 7.

Element Name	Details
`me:chebNumTemp`	Number of temperature points at which rate coefficients are to be calculated and used in the generation of the analytical representation.

me:chebNumConc	Number of concentration points at which rate coefficients are to be calculated and used in the generation of the analytical representation.
me:chebMaxTemp me:chebMinTemp	The maximum and minimum temperatures of the fit. The fit is undefined outside of this region. Execution will halt if me:chebMaxTemp $<$ me:chebMinTemp.
me:chebMaxConc, me:chebMinConc	The maximum and minimum bath gas concentrations of the fit. The fit is undefined outside of this region. The units attribute allows the concentration units to be specified and can be applied to either of these elements. Execution will halt if me:chebMaxConc $<$ me:chebMinConc.
me:chebTExSize	The number Chebyshev expansion points to be used in the temperature dimension. Execution will halt if me:chebTExSize $>$ me:chebNumTemp.
me:chebPExSize	The number Chebyshev expansion points to be used in the concentration dimension. Execution will halt if me:chebPExSize $>$ me:chebNumConc.

Table 7. Details of the Chebyshev representation parameters.

The rate coefficients of all reactions in the scheme will be fitted. The fits produced express the logarithm (base 10) of the rate coefficient as a function of the logarithm (base 10) of the concentration and the reciprocal temperature (see specific format documentation for details).

An example of the Plog representation specification is:

```
<me:calcMethod xsi:type="me:analyticalRepresentation">
  <me:format representation ="Plog" rateUnits="cm3mole-1s-1">cantera </me:format>
  <me:plogNumTemp>5</me:plogNumTemp>
  <!--this has a default-->
  <me:plogMaxTemp>1000</me:plogMaxTemp>
  <me:plogMinTemp>200</me:plogMinTemp>
  <me:plogConcs units="atm">
  <!--units has a default-->
        <me:plogConc>1</me:plogConc>
        <me:plogConc>2</me:plogConc>
        <me:plogConc>10</me:plogConc>
        <me:plogConc>100</me:plogConc>
  </me:plogConcs>
</me:calcMethod>
```

Note the attribute `representation` on the `me:format` element, which specifies the type of representation to be generated, the values being either "`Chebyshev`" or "`Plog`". The other child elements of the Plog representation are given in Table 8.

Element Name	Details
`me:plogNumTemp`	Number of temperature points at which rate coefficients are to calculated and used in the generation of the analytical representation.
`me:plogMaxTemp` `me:plogMinTemp`	The maximum and minimum temperatures of the fit. The fit is undefined outside of this region. Execution will halt if `me:plogMaxTemp` < `me:plogMinTemp`.
`me:plogConcs`	This element denotes the beginning of a list of pressures for each of which the modified Arrhenius parameters are to be determined. This element has an attribute, `units`, which specifies the units of pressure. (Note for ChemKin format only "`atm`" (atmospheres) is allowed.)
`me:plogConc`	This element specifies a pressure at which the modified Arrhenius parameters are to fit. Pressure are sorted into increasing order on entry.

Table 8. Details of the Plog representation parameters.

`ThermodynamicTable`: This class calculates the thermodynamic functions enthalpy (H(T)-H(0)), entropy (S(T)-S(0)), Gibbs energy (G(T)-G(0)), and constant pressure heat capacity C_p(T) of all the molecules defined in the molecule list of an input file at user defined temperatures. In addition, the heat of formation is output for those molecules that have a specified computational or thermodynamic energy, i.e. their input entry contains a `me:Hf0` or `me:Hf298` element, or a `me:ZPE` with an energy convention (see section 7.3.1.1). The minimum and maximum temperatures, the temperature interval and the units in which the thermodynamic functions are written can be specified as sub-elements, e.g.

```
<me:control>
  <me:calcMethod units="kJ/mol" xsi:type="me:ThermodynamicTable">
    <me:Tmin>75</me:Tmin>
    <me:Tmid>600</me:Tmid>
    <me:Tmax>1050</me:Tmax>
    <me:Tstep>75</me:Tstep>
  </me:calcMethod>
</me:control>
```

(This differs slightly from the earlier Mesmer versions.) 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. All of the input parameters have default values: `units="kJ/mol"`, `me:Tmin`=200, `me:Tmax`=1500, `me:Tstep` =50 < `me:Tmid` =1000; but these can be changed in defaults.xml. This method will attempt to process `molecule`s representing transition states as well and that their role be made clear by labelling them as `role="transitionState"`.

For those molecules with a computational or thermodynamic energy, a NASA polynomial (a representation of the temperature dependence of the thermodynamic properties) is automatically produced. The older (but more widely used) form is supported by Mesmer, and provides polynomial fits of C_p/R with 7 coefficients in two temperature ranges (separated by `me:Tmid`). Entropies, enthalpies and Gibbs energies can also be calculated from these coefficients, using standard thermodynamic relations. The XML output file contains both a standard fixed-format version of the polynomial and an XML representation. When this file is opened in Firefox (section 7.1) both the standard form (suitable for Chemkin) and a form suitable for Cantera are available for cutting and pasting. A detailed discussion of NASA polynomials and tabulations of coefficients for over 2500 species can be found at http://garfield.chem.elte.hu/Burcat/burcat.html. Note that the fifteenth coefficient is $\Delta_f H_{298.15K}/R$ in both the Burcat tabulation and the MESMER output.

Output of thermodynamic data can be added to any calculation by using an additional control block. In many cases the default parameters will be good enough and the control block can be:

```
<me:control>
  <me:calcMethod xsi:type="me:ThermodynamicTable"/>
</me:control>
```

as in examples/OH_NO_HONO_3Blks.xml.

Some minor additional features related to NASA polynomials are: a warning is given unless at least 6 temperatures in each range have been requested; if me:Tmid is 0, a single fit is produced and its coefficients are duplicated in the output; a fifteenth coefficient equal to the heat of formation at 298.15K divided by R is also produced.

It is often useful to be able to compare the thermodynamic functions that are produced by analytical formula with those based on calculating the canonical partition function from the density of states are the cell level, as this gives an indication of any errors that might be present in the density of states calculation. For this purpose, the child element has been added and when

specified will cause two columns to be written for each thermodynamic function in the .xml output, one for the analytical case and the other passed on density of states. The control sequence looks like:

```
<me:control>
   <me:calcMethod xsi:type="me:ThermodynamicTable"/>
   <me:withCellDOSCalc/>
</me:control>
```

and an example can be found in MesmerQA/ThermodynamicTable folder.

ErrorPropagation: This class performs a simple error estimate of the systems rate coefficients at specified temperature and pressure conditions based on a quasi-random sampling approach and the variance of the predicted outputs. This method requires a covariance matrix as part of the input, and so this method works best in conjunction with the Levenberg-Marquardt algorithm described above. The covariance matrix is used to generate Gaussian distributed parameter vectors, and then each of these vectors is sampled to calculate a distribution of predicted rate coefficients for the specified conditions, from which the standard deviation is calculated. Such errors are useful and can be used in the uncertainty and sensitivity analyses of large scale reaction mechanisms. Typically 2 or 3σ would be used in such studies.

The . xml input to use this method is:

```
<me:calcMethod xsi:type="me:ErrorPropagation">
   <me:errorPropagationSamples>512</me:errorPropagationSamples>
</me:calcMethod>
```

There is only one parameter at present, me:errorPropagationSamples, which specifies the number of Gaussian distributed parameter vectors to be generated in order to determine the variance. The default value (currently 256) from defaults.xml will be used if the element is omitted. It should be noted that the sample size required for error propagation tends to be smaller than that required for the global sensitivity analysis. For both methods, as a quasi-random sample using base 2 is used, optimal sample sizes will be 2^n e.g. 128, 256, 512, 1024. 128 samples would typically be sufficient for error propagation whereas for multi-parameter systems 1024 or larger may be required for the global sensitivity analysis.

The error estimates can be calculated in the same run as the fitting by having more than one control block (See examples/ErrorPropagation), shown here with default parameters:

```
<me:control>
   <me:calcMethod xsi:type="me:marquardt"/>
</me:control>
```

```xml
<!-- Conditions where error estimates required -->
<me:conditions>
  <me:bathGas>He</me:bathGas>
  <me:PTs>
    <me:PTpair units="PPCC" P="30.0e16" T="1000." precision="d/>
    <me:PTpair units="PPCC" P="3.00e16" T="1000." precision="d/>
  </me:PTs>
</me:conditions>

<!-- Error estimates for above conditions -->
<me:control>
  <me:calcMethod xsi:type="me:ErrorPropagation"/>
</me:control>
```

SensitivityAnalysis: This class is designed to perform a global sensitivity analysis of the predicted quantities against ME input parameters. The principal objective of sensitivity analysis is to identify those ME parameters that have the greatest impact on the uncertainty in predicted quantities. Consequently this information can be used to guide the choice of what parameters should be investigated in more detail in order to improve the robustness of the predictions. The sensitivity analysis implementation within MESMER is based on the Quasi-random Sampling High Dimensional Model Representation (QS-HDMR) approach developed in Ziehn and Tomlin [20] based on the earlier Monte Carlo based method of Li *et al.* [21]. In brief, the approach is based on the view that, for a given set of temperature and pressure conditions, the ME model is a device for mapping inputs, e.g. ME parameters such as barrier heights or $\langle \Delta E \rangle_d$, to outputs e.g. predicted rate coefficients where the inputs are subject to estimated uncertainties. For a given predicted quantity this mapping is represented as $f(\boldsymbol{x})$ where the vector $\boldsymbol{x}$ represents the input parameters which may vary over the specified input uncertainty range. The approach proceeds to construct a meta-model of this mapping by fitting a finite hierarchical correlated function expansion to input-output mappings obtained by running a large number of ME calculations, where for each run, the input parameters are sampled from their uncertainty distributions. In the case of uncorrelated parameters, uniform distributions are assumed for the input parameters with minimum and maximum values taken from the bounds set on the parameter as specified by the user (see section 8.5). The form of the hierarchical function expansion is as follows:

$$f(x) = f_0 + \sum_{i=1}^{n} f_i(x_i) + \sum_{1 \le i \le j \le n}^{n} f_{ij}(x_i, x_j) + \cdots + f_{12\cdots n}(x_1, x_{2,\cdots}, x_n) \qquad (11.1)$$

where *n* is the number of input parameters. Typically, for the purpose of sensitivity analysis, only terms up to second-order need be considered. This expansion is defined over a finite region

and the term f_0 represents the mean of the predicted outputs over the sample of ME simulations. The first-order terms represent the individual influence of input parameters on the uncertainty in predicted outputs. The higher order terms refer to progressively higher order interactions between input parameters in terms of their influence on predicted outputs. To progress the terms $f_i(x_i)$ are expressed as an expansion based on a set of orthogonal functions, φ_l:

$$f_i(x_i) = \sum_{l}^{m} \alpha_l^i \varphi_l(x_i) \tag{11.2}$$

the α_l coefficients of this expansion being determined by Monte-Carlo integration. Similarly, the terms $f_{ij}(x_i, x_j)$ are based on an expansion in the products of two orthogonal functions. The chosen sample size will influence the accuracy of the fitted expressions with an obvious computational penalty as the sample size is increased. It is the fitted expansion coefficients that yield information about the sensitivity of the output to the various inputs, the first-order sensitivity index of x_i, D_i , being given by:

$$D_i = \sum_{l}^{m} (\alpha_l^i)^2 / \sigma_f^2 \tag{11.3}$$

where σ_f^2 is the overall variance of the output calculated during the Monte-Carlo cycle.

The . xml input to use this method is:

```
<me:calcMethod xsi:type="me:sensitivityAnalysis">
  <me:sensitivityAnalysisSamples>400</me:sensitivityAnalysisSamples>
  <me:sensitivityAnalysisOrder>3</me:sensitivityAnalysisOrder>
  <me:sensitivityNumVarRedIters>10</me:sensitivityNumVarRedIters>
  <me:sensitivityVarRedMethod>RatioControl</me:sensitivityVarRedMethod>
</me:calcMethod>
```

Table 9 gives details of the specific keywords.

Element Name	Details
me:sensitivityAnalysisSamples	The number of quasi-random sample points to use in the Monte-Carlo integration to determine the orthogonal function expansion coefficients.
me:sensitivityAnalysisOrder	The order of the orthogonal function expansion to use.
me:sensitivityCorrelatedData	This element takes the values of true or false and indicates if the input parameters are correlated. This typically occurs when using the results from a Marquardt fit, which

	produces a correlation matrix. If the value is true MESMER will load the correlation matrix and use it to sample points from a Gaussian distribution during the Monte-Carlo integration.
`me:sensitivityVarRedMethod`	The variance reduction method to use. Variance reduction methods help to improve the accuracy of the Monte-Carlo integration and thus give better estimates of the sensitivity indices. There are at present two methods available: additive control, specifed by `AdditiveControl`, and ratio control specified by `RatioControl`.
`me:sensitivityNumVarRedIters`	The number of iterations to be used for the variance reduction cycle; typically, between 10 and 30.

Table 9. Details of the Sensitivity Analysis parameters.

The results from this analysis are presented in the XML output file, together with an estimate of the overall standard deviation in the predicted quantity (a quantity that might be used in sensitivity assessments of macroscopic simulations) and an R^2 assessment of the goodness of fit of the HDMR model. In general, the first and second order indices should sum to unity. However, if the fit is poor, that is if R^2 is significantly less that unity, the values of the second-order indices may be erroneous, and care should be taken in their interpretation.

The keyword `me:sensitivityCorrelatedData` needs a little more explanation – when specified this keyword will attempt to perform a sensitivity analysis that accounts for correlations between input variables. The relationship between the input variables must be specified as a correlation matrix, and for this reason this method should be used in conjunction with the Marquardt fitting method that generates such a matrix. Note that the indices from correlated inputs will depend on the order that variables appear in the correlation matrix. The resulting sensitivity indices therefore represent the marginal effects of parameters and care should be taken in terms of their interpretation. Readers are referred to Shannon et al. [22] for more details on this method.

11.2.2 Collisional Energy Transfer Models

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

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

$$P(E|E') = A(E')\exp\left(-\alpha(E' - E)\right) \tag{11.4}$$

where $A(E')$ is the normalization factor and the α parameter governs the size of energy transfer. α can be approximately associated with $1/\langle\Delta E\rangle_d$ where $\langle\Delta E\rangle_d$ is the average energy transferred in a deactivating collision. The probabilities of activating collisions are found by detailed balance.

It is often the case that $\langle\Delta E\rangle_d$ depends on other variables especially the temperature and sometimes the initial internal energy, E', of the reactant. In the MESMER implementation of the exponential down model these dependencies of $\langle\Delta E\rangle_d$ are modelled as:

$$\langle\Delta E\rangle_d = (\langle\Delta E\rangle_{d,ref} + \gamma E')\left(\frac{T}{T_{ref}}\right)^n \tag{11.5}$$

where γ is a constant, T is temperature, T_{ref} is the reference temperature, and n is an exponent governing the temperature dependence. By default, $\gamma = 0$ and $n = 0$, which means there is no temperature or internal energy dependency for $\langle\Delta E\rangle_d$, so that $\langle\Delta E\rangle_d = \langle\Delta E\rangle_{d,ref}$. The default for T_{ref} is 298 K. The syntax to specify the exponential down model is:

```
 <me:energyTransferModel xsi:type="me:ExponentialDown">
   <me:deltaEDown units="cm-1">210.0</me:deltaEDown>
   <me:deltaEDownTExponent referenceTemperature="298.">0.6</me:deltaEDownTExponent>
   <me:deltaEDownLinEne>0.0006</me:deltaEDownLinEne>
</me:energyTransferModel>
```

This gives a non-linear dependency of $\langle\Delta E\rangle_d$ on temperature with T_{ref} = 298, n = 0.6, and a linear dependence on E' with γ = 0.0006 and a $\langle\Delta E\rangle_{d,ref}$ = 210 cm^{-1}. The parameters $\langle\Delta E\rangle_{d,ref}$ and n can both be floated as part of a fitting exercise:

```
<me:energyTransferModel xsi:type="me:ExponentialDown">
   <me:deltaEDown units="cm-1" lower="140.0" upper="220." stepsize="10.0">
      210.0</me:deltaEDown>
   <me:deltaEDownTExponent lower="0.0" upper="1.0" stepsize="0.01">
      0.6</me:deltaEDownTExponent>
         <me:deltaEDownLinEne lower="1.e-06" upper="1.0" stepsize="1.e-06">
      0.0006</me:deltaEDownLinEne>
</me:energyTransferModel>
```

When analysing data from experiments using different bath gases, it is useful to be able to specify the energy transfer properties of more than one bath gas and this can be done by adding the name of the bath gas as an attribute, for example:

```
<me:energyTransferModel xsi:type="me:ExponentialDown">
  <me:deltaEDown bathGas="He" units="cm-1" lower="40.0" upper="100.0" stepsize="0.10">
54.1899</me:deltaEDown>
  <me:deltaEDownTExponent bathGas="He" lower="0.5" upper="1.5" stepsize="0.01">1.1749
</me:deltaEDownTExponent>
  <me:deltaEDown bathGas="H2" units="cm-1" lower="40.0" upper="400.0" stepsize="0.10">
46.9442</me:deltaEDown>
  <me:deltaEDownTExponent bathGas="H2" lower="0.5" upper="1.5" stepsize="0.01">1.41172
</me:deltaEDownTExponent>
  <me:deltaEDown bathGas="Ar" units="cm-1" lower="40.0" upper="400.0" stepsize="0.10">
73.8121</me:deltaEDown>
  <me:deltaEDownTExponent bathGas="Ar" lower="-1.5" upper="1.5" stepsize="0.01">0.00106756
</me:deltaEDownTExponent>
  <me:deltaEDown bathGas="ethane" units="cm-1" lower="75" upper="850" stepsize="0.10">
511.104</me:deltaEDown>
  <me:deltaEDownTExponent bathGas="ethane" lower="0.0" upper="1.5" stepsize="0.01">0.000993498
</me:deltaEDownTExponent>
</me:energyTransferModel>
```

BiExponentialDown: This calculates energy transfer probabilities based on the biexponential down model, an extension of the exponential down model. The probability of transition from a grain of energy E' to one of lower energy E is given by,

$$P(E|E') = A(E')[(1 - \gamma)\exp(-\alpha_1(E' - E)) + \gamma\exp(-\alpha_2(E' - E))] \qquad (11.6)$$

where $A(E')$ is the normalization factor, α_1 and α_2 are parameters that determine the size of the energy transfer and γ is a parameter that weights the contributions due to the two energy transfer parameters α_1 and α_2. When γ is zero, the biexponential form reduces to the exponential form and in this case α_1 can be associated with $1/\langle\Delta E\rangle_d$. Typically, $\alpha_1 > \alpha_2$. As before, the probabilities of activating collisions are found by detailed balance. The specification of this energy transfer model is:

```
<me:energyTransferModel xsi:type="me:BiExponentialDown">
   <me:deltaEDown units="cm-1">210.0</me:deltaEDown>
   <me:deltaEDown2 units="cm-1">500.0</me:deltaEDown2>
   <me:ratio>0.5</me:ratio>
</me:energyTransferModel>
```

where me:deltaEDown corresponds to $1/\alpha_1$, me:deltaEDown2 corresponds to $1/\alpha_2$ and me:ratio corresponds to γ. The parameters me:deltaEDown, me:deltaEDown2 and me:ratio can all be floated if required.

11.2.3 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, the Schrödinger equation for which is:

$$H\psi = -\frac{\hbar^2}{2}\frac{\partial}{\partial\theta}\frac{1}{I(\theta)}\frac{\partial\psi}{\partial\theta} + V(\theta)\psi = E\psi \tag{11.7}$$

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 identified 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:

$$V(\theta) = \sum_{n=0}^{M} V_n \cos n\theta \tag{11.8}$$

e.g. the simple potential (which might be used to model the rotation of a methyl group):

$$V(\theta) = \frac{V_0}{2}(1 - \cos 3\theta) \tag{11.9}$$

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` represents the value of the non-zero coefficients n in the cosine expansion (11.8) and `coefficient` the associated cosine expansion coefficient.

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:

```xml
<me:ExtraDOSCMethod xsi:type="me: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 >
```

For many internal rotors the reduced moment of inertia is also a function of the angle defining the internal rotation. This occurs if one of the fragments that define the internal rotation is asymmetric with respect to this rotation. This causes kinematic coupling between the internal rotors and between internal rotation and the overall rotation. To account for all the couplings is, in general, a complex problem, but some of these effects can be accounted for in the one-dimensional case by solving the Schrodinger equation with a reduced moment of inertia that depends upon the angle of internal rotation, $I(\theta)$. The function $I(\theta)$ can be defined in one of two ways: it can be entered as a Fourier cosine expansion similar to that for the potential e.g.

```xml
<me:InternalRotorInertia units="amuA^2">
  <me:InertiaPoint index="0" coefficient="11.831"/>
  <me:InertiaPoint index="1" coefficient="-4.9066"/>
  <me:InertiaPoint index="2" coefficient="-3.1732"/>
  <me:InertiaPoint index="3" coefficient="-0.46872"/>
  <me:InertiaPoint index="4" coefficient="1.0715"/>
  <me:InertiaPoint index="5" coefficient="0.74663"/>
  <me:InertiaPoint index="6" coefficient="-0.19779"/>
  <me:InertiaPoint index="7" coefficient="-0.42143"/>
</me:InternalRotorInertia>
```

Care should be taken when entering these coefficients that they in phase with the hindering potential. Alternatively, if there are atomic coordinates available for the species MESMER can calculate $I(\theta)$ (or, more precisely, $1/I(\theta)$) by constructing the combined internal and external rotation kinetic energy tensor (including Coriolis terms) followed by inversion and extraction of the appropriate term. This method can be used by specifying the keyword:

```xml
<me:CalculateInternalRotorInertia phaseDifference ="180.0"/>
```

This method is demonstrated in the Butyl_H_to_Butane example (See examples table in Section 10.2). The attribute `phaseDifference` accounts for the difference in angle uses to define the atomic configuration and that used for the potential. The $1/I(\theta)$ term is expanded as Fourier series of cosine functions and, if the attribute `UseSineTerms="yes"`, sine terms also, though experience to date suggests these latter terms make a minor contribution.

Some internal rotations impose 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.

It is important to make sure that if a hindered rotor is specified that a corresponding vibrational frequency is removed from the array of vibrational frequencies in order that the number of degrees of freedom has the correct value. The frequency in question can be removed from array or it can be referred to as part of the definition of the hindered rotor:

`<me:replaceVibFreq>543</me:replaceVibFreq>`

Determination of which frequency to eliminate can be a difficult process, and it is not safe to assume that the lowest frequencies correspond to internal rotation. If a Hessian is available then MESMER will use the projection method reported by Sharma *et al*. [23] to project out the mode associated with internal rotation.

WARNING: When calculating frequencies from a Hessian, including the case when internal rotor degrees of freedom are projected out, it is critical that the Hessian and species coordinates presented to MESMER are expressed relative to the same Cartesian coordinate system.

In some cases, it may be desirable to project out a hindered rotor mode but not include the states of the mode in the density of states calculation. For example, an isomerization that takes place via a cyclic transition state may require certain internal rotors to have specific values for the reaction to take place, other configurations not leading to reaction. An approximate why to deal with this is to fix the geometry for the transition state, and remove the motion associated with

the hindered rotors that define the transition state configuration. This can be done by applying the keyword `me:TSExclusion` to the definition of the hindered rotor. This ensure that it will be project out of any Hessian but that the associated states will not be included in the density of states. An example of the use of this keyword can be found in `…\MesmerQA\pentyl\` `pentyl_isomerization_restrainedTS.xml`.

More detail of the calculation of the energy levels of an internal rotor can be found in ref [1].

`ClassicalCoupledRotors`: this class should be used when there is significant coupling between internal and external rotations. Typically, this occurs for molecules which have a large number of internal rotations, for example the higher alkanes, where, in addition to Coriolis coupling, the moment of inertia tensor will depend on the internal rotational degrees of freedom, so further coupling internal and external rotation. The theory underlying this method is described in section 14.4. Because there is coupling between internal and external rotation, this method is designated as `<me:DOSCMethod>`, that is, to use this method it must be the only `<me:DOSCMethod>` defind. Also, for this method to be used the geometry of the lowest energy configuration must be specified, including all bonds about which rotation can occur. The definition of a coupled system looks like (see the input files in the CoupledRotors example folder for illustrations of this method (see also examples table in Section 10.2)):

```
<me:DOSCMethod xsi:type="me:ClassicalCoupledRotors">
  <me:MCPoints>1000</me:MCPoints>
  <me:RotorArray>
    <me:Rotor bondRef="b1" periodicity="3">
      <me:HinderedRotorPotential format="analytical" units="cm-1">
        <me:PotentialPoint index="0"  coefficient="255.0"/>
        <me:PotentialPoint index="3"  coefficient="-255.0"/>
      </me:HinderedRotorPotential>
    </me:Rotor>

    <me:Rotor bondRef="b2" periodicity="3">
      <me:HinderedRotorPotential format="analytical" units="cm-1">
        <me:PotentialPoint index="0"  coefficient="255.0"/>
        <me:PotentialPoint index="3"  coefficient="-255.0"/>
      </me:HinderedRotorPotential>
    </me:Rotor>
  </me:RotorArray>
</me:DOSCMethod>
```

The element `<me:MCPoints>` specifies the number of points to use for the Monte-Carlo integration. The element `<me:RotorArray>` specifies the beginning of an array of internal rotor definitions. The internal rotor definition `<me:Rotor bondRef="b2" periodicity="3">` has attributes that identify the bond about which internal rotation occurs and the periodicity of that

rotation. The hindering potential can be defined as a child element of `me:Rotor` and has the same structure as that given for `HinderedRotorQM1D` above.

`Morse`: this class calculates the energy levels of a collection of decoupled Morse oscillators:

$$V(R) = D[1 - \exp{(-\alpha(R - R_e))}]^2 \tag{11.10}$$

where D is the dissociation energy and R_e is the equilibrium value of the coordinate R. The energy levels of an individual oscillator are given by:

$$E(n) = \left(n + \tfrac{1}{2}\right)h\upsilon_0 - \tfrac{1}{4D}\left[\left(n + \tfrac{1}{2}\right)h\upsilon_0\right]^2 = \left(n + \tfrac{1}{2}\right)\bar{\nu} - \left(n + \tfrac{1}{2}\right)^2\omega_e\bar{\nu} \tag{11.11}$$

where υ_0 is the given by

$$\frac{a}{2\pi}\sqrt{2D/m} \tag{11.12}$$

m being the associated mass. The collection is specified as follows:

```
<me:ExtraDOSCMethod xsi:type="Morse">
  <me:MorseParameters vibrationalFrequency="3161.925" anharmonicity="-37.593"/>
  <me:MorseParameters vibrationalFrequency="3062.283" anharmonicity="-17.944"/>
      ...
  <me:MorseParameters vibrationalFrequency="1901.381" anharmonicity="-11.53"/>
  <me:MorseParameters vibrationalFrequency="1460.162" anharmonicity="-5.239"/>
</me:ExtraDOSCMethod >
```

The attribute `vibrationalFrequency` is equal to $\bar{\nu}$ (υ_0) and the attribute `anharmonicity` is equal to $-\omega_e\bar{\nu}$.

`DefinedDensityOfStates`: Occasionally, it is desirable to calculate or manipulate densities of states outside of MESMER. This class allows such densities of states to be read in to MESMER and then used in the usual way. The input for this method looks like this:

```
<me:DOSCMethod>DefinedDensityOfStates</me:DOSCMethod>
<me:States units="cm-1" grainSize ="100.0">
  <me:State energy="0.0"      degeneracy="0.0"/>
  <me:State energy="61.0994"  degeneracy="2123800"/>
  <me:State energy="160.991"  degeneracy="8647700"/>
  <me:State energy="258.927"  degeneracy="25886200"/>
  <me:State energy="357.325"  degeneracy="70904000"/>
    ...
  <me:State energy="148250"   degeneracy="3.85697E+77"/>
  <me:State energy="148350"   degeneracy="4.0407E+77"/>
  <me:State energy="148450"   degeneracy="4.23308E+77"/>
  <me:State energy="148550"   degeneracy="4.4345E+77"/>
</me:States>
```

This method differs from the others in that it replaces all other state calculators, hence this method is designated a `<me:DOSCMethod>`. The data input is usually manipulated as follows: the log of the degeneracy is taken (this can be supressed if desired by specifying the attribute

`nologSpline`) and the resulting data set is then fitted to a spline. The spline is then used to calculate the degeneracy value for cells of specified size. If the degeneracies entered are the number of states per cm^{-1} at the specified energy then `grainSize` does not have to be specified, otherwise the value of the grain width used, in the same units as the energy, must be specified.

`FourierGrid1D`: This class calculates the solutions of the one-dimensional Schrödinger equation for an arbitrary potential, that is solutions of,

$$H\psi = -\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2} + V(x)\psi = E\psi \tag{11.13}$$

using the methods described by Marston and Balint-Kurti et al.[24] In this approach, the basis set is composed of a set of δ-functions at various coordinate values such that the potential is given by,

$$V(x) = \sum_i^N V(x_i)\delta(x - x_i) \tag{11.14}$$

and kinetic energy matrix elements are then similar expressed in terms of this basis set. The potential can be input in one of two ways: as an `analytical` represented as a polynomial, for example,

```
<me:vibrationalPotential format="analytical" units="kJ/mol" minx="0.6" maxx="1.8">
    <me:PotentialPoint index="0"  coefficient="5602.47"/>
    <me:PotentialPoint index="1"  coefficient="-9312.16"/>
    <me:PotentialPoint index="2"  coefficient="3869.56"/>
</me:vibrationalPotential>
```

where `index` is the power of the polynomial term and `coefficient` the associated polynomial coefficient. Alternatively, the hindering potential can be expressed in `numerical` format, as a set of potential points at regular intervals of the coordinate, for example,

```
<me:vibrationalPotential format="numerical" units="kJ/mol" minx="0.6" maxx="1.8">
    <me:PotentialPoint coordinate="0.1"  potential="4709.9496"/>
    <me:PotentialPoint coordinate="0.2"  potential="3894.8204"/>
    <me:PotentialPoint coordinate="0.3"  potential="3157.0824"/>
        ...
    <me:PotentialPoint coordinate="2.1"  potential="3111.6936"/>
    <me:PotentialPoint coordinate="2.2"  potential="3844.3884"/>
    <me:PotentialPoint coordinate="2.3"  potential="4654.4744"/>
</me:vibrationalPotential>
```

where `coordinate` is the value of the coordinate and `potential` the value if the potential associated with that coordinate value. The values input for this potential will be used to construct a spline representation of the potential and these will be used to generate the potential values at the required grid points.

In both cases the `units` in which the potential is measured can be specified and the default is kJ/mol. Also, specified in this section are the range of coordinate values to be used in the calculation of the energy levels, `minx` being the minimum value of the coordinate and `maxx` being the maximum value. This range should encompass all significant minima and as much of the confining potential as is consistent with the maximum energy in the overall calculation. These values are read in at this point so that they can be checked against the range of values supplied for potential coordinate.

An example of the full specification of this calculator is:

```
<me:ExtraDOSCMethod xsi:type="me:FourierGrid1D">
  <me:NumGridPnts>100</me:NumGridPnts>
  <me:reducedMass>7.9997</me:reducedMass>
  <me:replaceVibFreq>1645.28</me:replaceVibFreq>
  <me:vibrationalPotential format="analytical" units="kJ/mol" minx="0.6" maxx="1.8">
    <me:PotentialPoint index="0" coefficient="5602.47"/>
    <me:PotentialPoint index="1" coefficient="-9312.16"/>
    <me:PotentialPoint index="2" coefficient="3869.56"/>
  </me:vibrationalPotential>
</me:ExtraDOSCMethod>
```

The other elements specify the following: the element `me:NumGridPnts` specifies the number of grid points to be used for the solution and corresponds to the number of intervals into which the range `minx` to `maxx` is divided. It is also the size of the Hamiltonian matrix that is constructed. It must be even and if an odd number is entered it is set to the next highest even number and has a default value of 100.

A reduced (or effective) mass must be specified, and this is done in one of two ways: using the element,

```
<me:reducedMass>7.9997</me:reducedMass>
```

Some care is required in the determination of this value and particular attention to the units used is also required. By default, the units are amu. This default assumes that the motion is linear and that the displacements `minx` and `maxx` expressed in Å. However, if the motion being modelled is a bending vibration the units of displacement are best represented as radians (that is effectively dimensionless) and the units of reduced mass should be in amuÅ^2.

Alternatively, if the coordinates of the fragment are available the reduced mass may be calculated. To do this the bonds that defined the bend have to be specified and this can be done using the element,

```
<me:bondRef>b1 b2</me:bondRef>
```

The definitions of these bonds must be such that they share a common atom which will be the hinge point of the bend and from this definition the reduced mass is calculated for the geometry presented. (It should be noted that, as with hindered rotation, the reduced mass may also be a function of the angular coordinate used to define the mode, however, at present this extension has not been implemented.) It is also possible to calculate the reduced mass for an out of plane bend (or wag) and this is done with,

```
<me:bondRef>b1 b5 b6</me:bondRef>
```

where each of the three bond definitions must share a common atom, the inversion centre, the first bond is to the atom or group that wags and the plane against which the first bond wags is defined by the last two bonds.

Some modes have associated with them a symmetry, for example the inversion mode of ammonia. It is important to account for this symmetry and this can be done by combining this mode symmetry with the overall rotation symmetry or, more explicitly, with an element of the form:

```
<me:symmetryNumber>2</me:symmetryNumber>
```

As with the hindered rotor case it is important to make sure that, if a `me:FourierGrid1D` term is specified, that a corresponding vibrational frequency is removed from the array of vibrational frequencies in order that the number of degrees of freedom has the correct value. Again, this term done using an element of the form:

```
<me:replaceVibFreq>1645.28</me:replaceVibFreq>
```

Examples of this this method in use can be found in the file `ThermodynamicTable_Fourier.xml` of the `MesmerQA\ThermodynamicTable` folder.

Note: The approach used here is conjugate to the method used for hindered rotors above, in the sense that the basis functions used for hindered rotors method are a set of δ-functions in momentum space. While it is possible to use this approach for hindered rotors as well, the strict periodic nature of hindered rotors, the sometimes complex (in the sense of having real and imaginary parts) of the hindered rotor wavefunctions, and the dependence of the hindered internal rotor moment of inertia on the internal rotor angle (see Eq. 11.6), it is recommended that the `HinderedRotorQM1D` be used for specific case of hindered rotors.

11.2.4   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:

(1) `RRKM`: uses the standard, well known RRKM expression to calculate $k(E)$. This method requires a `me:transitionState` to be specified. A typical specification is:

```
<reaction id="R2" >
  . . .
  <me:transitionState>
    <molecule ref="TS2" role="transitionState" />
  </me:transitionState>
  <me:MCRCMethod name="RRKM"/>
</reaction>
```

(2) `MesmerILT:` If no transition state is available, then `MesmerILT` may be used. The inverse Laplace transform (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 or modified Arrhenius fit to a set of limiting high pressure canonical rate coefficients, $k_\infty(T)$, for dissociation, isomerization or association, which may be obtained from experiment or theory. The mathematical details can be found in section 14.3. However, it is important to stress that the quality of the $k(E)$s obtained via this procedure are very dependent on the quality of the Arrhenius parameters. Typically, the greater the barrier to reaction, the greater the range of temperatures the measured $k_\infty(T)$ must cover to be confident of the Arrhenius parameters and, consequently, the $k(E)$s. Our recommendation is that this technique should be used for association reactions only. If `MesmerILT` is specified, then three parameters from a modified Arrhenius fit to $k_\infty(T)$ are required: $k_\infty(T) = A_\infty \left(\frac{T}{T_\infty}\right)^{n_\infty} \exp\left(-E_\infty/RT\right)$ 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.) Optionally, the reference temperature, T_∞, can also be specified (`me:TInfinity`) but cannot be floated in a fitting exercise. An example specification is:

```
<me:MCRCMethod xsi:type="me:MesmerILT">
```

```
   <me:preExponential units="cm3molecule-1s-1" lower="1e-13" upper="9e-11"
stepsize="1e-13">10.61e-12</me:preExponential>
   <me:activationEnergy units="kJ/mol">0.0</me:activationEnergy>
   <me:TInfinity>298.0</me:TInfinity>
   <me:nInfinity lower="-1.0" upper="1.0" stepsize="0.005">-0.6</me:nInfinity>
 </me:MCRCMethod>
```

The ILT methods can be used to fit association rate data, reactions with bimolecular source, or sink terms, typically involving barrierless potential energy surfaces. Acetyl + O_2, which is discussed in section 10.1.5, provides an example of the former, which, in MESMER, is termed the standard ILT. In this case, $k(E)$ for dissociation of $CH_3C(O)O_2$ to $CH_3CO + O_2$ is calculated from the association $k_\infty(T)$ for the reaction of $CH_3CO + O_2$, the Arrhenius parameters being obtained from the from the literature. Alternatively, if MESMER is used to fit experimental rate data for $CH_3CO + O_2$, then these parameters can be returned from the fitting process. These parameters are specified in the input file as:

```
<me:MCRCMethod xsi:type="MesmerILT">
   <me:preExponential units="cm3molecule-1s-1">6.00e-12</me:preExponential>
   <me:activationEnergy units="kJ/mol">0.0</me:activationEnergy>
   <me:TInfinity>298.0</me:TInfinity>
   <me:nInfinity>-0.5</me:nInfinity>
</me:MCRCMethod>
```

The dissociation of I_2O_2, formed from the reaction IO + IO, provides an example of dissociation to form two sink molecules, OIO + I. In MESMER, this is termed a reverse ILT; the Arrhenius parameters for the OIO + I association reaction are used to determine the dissociation $k(E)$ even though this reverse reaction is not included in the final reaction scheme. 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" role="modelled" />
  </reactant>
  <product>
   <molecule ref="OIO" role="sink" />
  </product>
  <product>
   <molecule ref="I" role="sink" />
  </product>
  <me:MCRCMethod xsi:type="MesmerILT">
    <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>
  </me:MCRCMethod>
</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, that is the association activation barrier. The input Arrhenius data are for the OIO + I association reaction, but

`reverse="true"` tells MESMER to use those 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 are 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 are essential, and MESMER will terminate if they are not provided. A further example of the use of the reverse ILT can be found in 12.1.6 for i-propyl dissociation.

For dissociation and isomerisation reactions, ILT provides a means of rapidly estimating $k(E)$, without recourse to RRKM calculations and the required transition state parameters. If an experimental expression for $k_\infty(T)$ is available, then $k(E)$ can be estimated simply from a knowledge of the reactant densities of states, as demonstrated in section 14.3.1. As emphasised above, the quality of the resulting microcanonical rate coefficients depends sensitively on the accuracy of the expression for $k_\infty(T)$.

Sometimes, a single Arrhenius form is not sufficient to represent the temperature dependence of the rate coefficient. This might be because there are two similar transition states on the reaction surface or the microcanonical rate coefficients have a complex dependency on internal energy. In these cases, it can be useful to use more than one Arrhenius form, such that:

$$k(\beta) = A_{\infty,1} \left(\frac{\beta_\infty}{\beta}\right)^{n_1} \exp(-\beta E_{\infty,1}) + A_{\infty,2} \left(\frac{\beta_\infty}{\beta}\right)^{n_2} \exp(-\beta E_{\infty,2}) \qquad (11.15)$$

where $\beta_\infty = 1/kT_\infty$, the reciprocal reference temperature, is the same for both terms. This form can be specified in a MESMER input file as:

```
<me:MCRCMethod xsi:type="me:MesmerILT">
  <me:activationEnergy units="kJ/mol">5.375</me:activationEnergy>
  <me:preExponential units="cm3molecule-1s-1">7.00e-12</me:preExponential>
  <me:nInfinity>0.0</me:nInfinity>
  <me:secondaryPreExponential units="cm3molecule-1s-1">15.00e12
</me:secondaryPreExponential>
  <me:secondaryActivationEnergy units="kJ/mol">15.375
</me:secondaryActivationEnergy>
  <me:secondaryNInfinity>0.5</me:secondaryNInfinity>
</me:MCRCMethod>
```

Finally, due to the mathematics of the ILT, it is necessary to point out that there are some constraints on the parameters: for the standard and reverse ILT, n_∞ must be greater than -1.5, while for the unimolecular ILT, n_∞ must be greater than zero. If a zero n_∞ is desired, setting it to a very small number will introduce minimal error.

(3) `LandauZenerCrossing`: this method extends the method `RRKM` to allow for spin forbidden reactions (surface hopping or intersystem crossing). The details of this method are described by Harvey and Aschi [16]. If the molecule being modelled is a minimum energy crossing point (MECP) – i.e., a transition state for spin forbidden crossing, then further data are required: `me:RMS_SOC_element` the root mean squared spin orbit coupling element, specified following with `units="cm-1"`; `me:GradientReducedMass` the reduced mass for movement along the direction orthogonal to the crossing seam, specified following with `units="a.m.u."` and `me:GradientDifferenceMagnitude` the norm of the vector representing the gradient difference at the MECP, specified following with `units="a.u./Bohr"` An example of the required syntax can be found in `examples\spin_forbidden_kinetics\LZ_test.xml`, and has the following form:

```
<me:MCRCMethod name="LandauZenerCrossing" xsi:type="LandauZenerCrossing">
    <me:RMS_SOC_element units="cm-1">5.08</me:RMS_SOC_element>
    <me:GradientDifferenceMagnitude units="a.u./Bohr">0.09327305
</me:GradientDifferenceMagnitude>
    <me:GradientReducedMass units="a.m.u.">10.4098</me:GradientReducedMass>
 </me:MCRCMethod>
```

(4) `WKBCrossing`: This method is similar to the `LandauZenerCrossing` method, but includes tunnelling corrections below threshold and is also. described by Harvey and Aschi [16]. WKB transmission probabilities require one additional input: `me:AverageSlope` the geometric mean of the norms of the gradients on the two surfaces at the MECP with `units="a.u./Bohr"` An example of the required syntax can be found in `examples\spin_forbidden_kinetics\WKB_test.xml`.

(5) `ZhuNakamuraCrossing`: This method, based on the work of Zhu and Nakamura [18], is another approach to the calculation of intersystem crossing rate coefficients. In the present implementation, the diabatic reactant surface is modelled as a parabolic potential, $V_r = k_f(x - x_0)^2 + E_r$, and the diabatic product surface as a dissociative exponential potential, $V_p = Ae^{Bx} + E_p$, both functions of the reaction coordinate x. The reactant surface is expressed in terms of three parameters: `<me:harmonicReactantDiabat-X0>`, the equilibrium position along the reaction coordinate of the reactant, x_0, in units of Bohr; `<me:harmonicReactantDiabat-FC>`, the force constant or steepness of the parabolic potential, k_f, in kJ/mol/Bohr^2; `<me:harmonicReactantDiabat-DE>`, the energy, in kJ/mol, of the reactant at equilibrium, E_r. The product surface is also expressed in terms of three parameters:

111

`<me:exponentialProductDiabat-A>`, the pre-exponential factor, A, in kJ/mol; `<me:exponentialProductDiabat-B>`, the parameter B in Bohr^{-1}, which is negative and governs the repulsive character of the product surface; `<me:exponentialProductDiabat-DE>` the effective dissociation energy, E_p, in kJ/mol, of the product surface. These two surfaces are coupled by the off-diagonal matrix element `<me:H12>` which has units of kJ/mol. Finally, and in common with other methods, a reduced mass is required and this is specified as `<me:reducedMass>`. A complete specification of this method looks like (see examples\spin_forbidden_kinetics\ZN_test.xml):

```
<me:MCRCMethod name="ZhuNakamuraCrossing" xsi:type="ZhuNakamuraCrossing">
   <me:harmonicReactantDiabat-FC units="kJ/mol/Bohr**2">166.614
</me:harmonicReactantDiabat-FC>
   <me:harmonicReactantDiabat-X0 units="Bohr">-0.487</me:harmonicReactantDiabat-X0>
   <me:harmonicReactantDiabat-DE units="kJ/mol">0.0</me:harmonicReactantDiabat-DE>
   <me:exponentialProductDiabat-A units="kJ/mol">89.86</me:exponentialProductDiabat-A>
   <me:exponentialProductDiabat-B units="1/Bohr">-3.74</me:exponentialProductDiabat-B>
   <me:exponentialProductDiabat-DE units="kJ/mol">-46.34</me:exponentialProductDiabat-DE>
   <me:H12 units="kJ/mol">7.00</me:H12>
   <me:reducedMass units="a.m.u.">9.072</me:reducedMass>
 </me:MCRCMethod>
```

The Zhu-Nakamura approach includes tunnelling.

(6) `DefinedSumOfStates`: This class has been implemented in order to allow transition state sums of states, $W(E)$ or $W(E, J)$ where E is energy and J total angular momentum, calculated by other means, to be input into MESMER. An example might be to input the transition state sums of states of a reaction where there is no well-defined location of the transition state, thus requiring the use of a variational technique such as FTST to locate the transition state. This method requires a transition state species to be defined in the molecule list and this definition must include the sum of state in the following format:

```
<me:MCRCMethod xsi:type="DefinedSumOfStates">
 <me:SumOfStates units="cm-1" angularMomentum="true" noLogSpline="false">
  <me:SumOfStatesPoint energy="154.87" angMomMag="0">0.1262E+02</me:SumOfStatesPoint>
  <me:SumOfStatesPoint energy="247.91" angMomMag="0">0.2401E+02</me:SumOfStatesPoint>
…
  <me:SumOfStatesPoint energy="19439.49" angMomMag="80">0.657E+08</me:SumOfStatesPoint>
  <me:SumOfStatesPoint energy="22216.56" angMomMag="80">0.276E+09</me:SumOfStatesPoint>
 </me:SumOfStates>
</me:MCRCMethod>
```

The tag `me:SumOfStates` indicates the beginning of a sum of states definition and allows a number of other parameters to be set: `units` allows the units of E to be specified. `angularMomentum` indicates if the sum of states has an angular dependence i.e. $W(E, J)$ (`true`) rather than $W(E)$ (`false`). The energies for which the sum of states are specified are unlikely to correspond to the division chosen for the grain boundaries of the ME

calculation, so splines are used to interpolate the sum of states at a given point. Typically, the sum of states can vary greatly and so it is usually better to perform the interpolation using $lnW(E,J)$ or $lnW(E)$ and this is the default, however in some cases it may be better to use the sum of state as entered and the attribute `noLogSpline` allows for this. The `me:SumOfStates` tag is followed by a series of `me:SumOfStatesPoint` tags that specify a point that defines the $W(E)$ or $W(E,J)$ function. The attributes of this tag specify the energy (which must be present) and the angular momentum (which must be present only be present if `angularMomentum` is `true`) for which a sum of states is specified. An example of the use of this method can be found in `examples\Methyl_H_to_Methane\Methyl_H_to_Methane_FTST`.

(7) `CanonicalRateCoefficient`: Strictly speaking, this class in not a microcanonical rate coefficient class, but is included here for completeness. The class generates canonical rate coefficients and is designed for use with the irreversible exchange reaction (see section 7.3.2), to allow for bimolecular loss of one of the reactants of the system due to a side reaction, the side reaction being assumed to proceed via a regular, bimolecular transition state. The definition of this method is:

```
<me:MCRCMethod xsi:type="me:CanonicalRateCoefficient">
 <me:preExponential units="cm3molecule-1s-1" lower="1.E-13" upper="2.0E-10"
stepsize="1.0E-13" >2.42095e-11</me:preExponential>
 <me:activationEnergy units="kJ/mol" lower="15.0" upper="30.0" stepsize="0.5">23.5312
</me:activationEnergy>
 <me:TInfinity>1.0</me:TInfinity>
 <me:nInfinity>0.0</me:nInfinity>
</me:MCRCMethod>
```

The definition specifies that canonical Arrhenius form to be used for the reaction rate coefficient. As shown in this example, the parameters of this method can also be treated as fitting parameters, which is useful in a fitting exercise when the rate of the side reaction may not be well defined.

11.2.5 Tunneling Corrections

The abstract base class for tunnelling corrections is `TunnelingCalculator`, which lives in `Tunneling.h`. Note that MESMER commands/files use the American spelling – **tunneling**. The derived concrete classes (i.e., the plug-in classes) offer different methods for calculating tunnelling corrections as follows:

`EckartCoefficients`: this class approximates tunneling for a one dimensional asymmetric Eckart potential using the method described by Miller [4] and specified using the `Eckart` type, for example,

```
<me:tunneling xsi:type="Eckart"/>
```

in the section(s) defining the reaction(s) to which tunnelling is to be applied. An optional element, `me:useZPE`, can be specified that accounts for the zero-point energy, that is the forward and reverse barriers are adjusted for zero-point energy, for example,

```
<me:tunneling xsi:type="Eckart">
   < me:useZPE>
</me:tunneling>
```

If this element is not specified, the classical barriers are used. Alternatively, the barrier heights can be specified directly,

```
<me:tunneling xsi:type="Eckart">
   <me:BarrierHeights units="cm-1" V0="9864." V1="8833."/>
</me:tunneling>
```

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>
```

`WKBTunneling`: This class approximates tunnelling using a user defined potential according to a semi-classical 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 relevant transition state. This potential is then numerically integrated to give the barrier penetration integral ϕ as described by Garrett and Truhlar.[5] This method is specified in the section(s) defining the reaction(s) to which tunnelling is to be applied as follows::

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

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.

There is an example in `examples\Tunnelling\H+H2,T+T2.xml`.

DefinedTunnelingCoefficients: This class reads tunneling coefficients calculated from some external source. This class requires tunnelling probabilities as a function of energy (presently the energy units must be cm^{-1}) and the zero of energy should correspond to the ZPE of the reactant (s) specified in the reaction elements. These discrete tunnelling probabilities are interpolated over the full energy range required using a spline function. The method is specified in the section(s) defining the reaction(s) to which tunnelling is to be applied as follows:

```
<me:tunneling xsi:type="Defined">
  <me:DefinedTunnelingCoefficients>
    <me:DefinedPE energy= "0.00"    pE= "3.96E-15"/>
    <me:DefinedPE energy= "1.89"    pE= "4.09E-15"/>
    . . .
    <me:DefinedPE energy= "2542.33"  pE= "4.98E-01"/>
    <me:DefinedPE energy= "2544.185" pE= "5.00E-01"/>
  </me:DefinedTunnelingCoefficients>
</me:tunneling>
```

Where Energy is relative to the reactant ZPE in cm^{-1} and pE is the corresponding tunnelling probability at this energy.

There is an example in `examples\DefinedTunellingCoefficients\OH+methanol.xml`.

11.2.6 Distribution Calculator

The abstract base class for calculating the initial distributions within the grains is `DistributionCalculator`, which is located in `Distribution.h`. Note, rate coefficients calculated using the Bartis-Widom method are independent of the initial conditions. The derived concrete classes (i.e., the plug-in classes) offer different methods for the initial grain distribution as follows:

`Boltzmann`: This is the default distribution and calculates a Boltzmann population for the initial grain distribution within a particular well. It allows a temperature to be specified, which can be used in the simulation of shock tube experiments. If the temperature is not specified, the current simulation temperature is assumed. An example of the specification of the distribution is,

```
<me:DistributionCalcMethod xsi:type="me:Boltzmann">
  <me:Temperature>600.0</me:Temperature>
</me:DistributionCalcMethod>
```

In addition to specifying the temperature it is also possible to specify an excitation that shifts the entire energy distribution up fixed energy. This is useful as an approximate model of a photo-excitation experiment, for example,

```
<me:DistributionCalcMethod xsi:type="me:Boltzmann">
   <me:Temperature>300.0</me:Temperature>
   <me:Excitation units ="cm-1">1000.0</me:Excitation>
</me:DistributionCalcMethod>
```

The attribute `units` on the child element `me:Excitation` specifies the energy units that the excitation energy are specified in. An example of the use of the `me:Excitation` keyword can be found in `…\examples\cis_to_trans_But-2-ene\cis_to_trans_But-2-eneEx.xml`.

`Prior`: This distribution approximately models the distribution that a species emerging from a dissociation process might have. For the dissociation

$$A(E_x) \longrightarrow B + C \tag{11.15}$$

Where the species A has a well-defined excess energy E_x, i.e. energy in excess of the reaction threshold, the prior distribution of the species B is given by,

$$p_B(E|E_x) = \frac{N_B(E)[N_t \otimes N_C](E_x - E)}{[N_B \otimes N_t \otimes N_C](E_x)} \tag{11.16}$$

where $p_B(E)$ is the probability of that species B emerges from the dissociation with energy E, $N_B(E)$ is the rovibrational density of states of species B, $N_C(E)$ is the rovibrational density of states of species C, is the relative translational density of states of dissociating species B and C, for which a classical expression is used, that is $N_t(E) \propto \sqrt{E}$. The notation $[N_X \otimes N_Y](E)$ represents a convolution:

$$[N_X \otimes N_Y](E) = \int_0^E N_X(E - z)N_Y(z)dz \tag{11.17}$$

Note that the range of the energy E is $0 \le E \le E_x$, outside this range $p_B(E) = 0$. Typically, this distribution will be useful when species B, prepared by a dissociation process at a well-defined excess energy, reacts with other species and it these subsequent reactions that are of interest. The distribution is specified by,

```
<me:DistributionCalcMethod xsi:type="me:Prior">
  <me:CoFragment>I</me:CoFragment>
  <me:EnergyExcess units="kJ/mol">50</me:EnergyExcess>
</me:DistributionCalcMethod>
```

The two child elements are: `CoFragment`, this specifies the identity of the other species emerging from the dissociation and must be specified in the usual way in the molecule; `EnergyExcess`, this specifies the excess energy of the initial adduct and is equal to E_x above.

There is an example of this method in `MesmerQA\pentyl\pentylPrior.xml`.

11.2.7 Fragmentation Distribution

located in `PseudoIsomerizationReaction.h`. These distributions are used in conjunction with the pseud-isomerization reaction type (see section 7.3.2), for the situation where the deficient species of an association reaction has a non-Boltzmann distribution, typically this occurs when the associating species is a product of a unimolecular sequence. Consider reaction (11.12) written in the reverse sense:

$$B(E_b) + C \rightleftharpoons A(E_x)$$ (11.18)
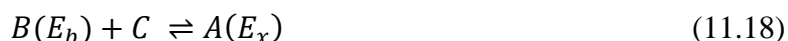
Where C is in excess. The quantity of interest is the probability of B emerging from the dissociation with energy E_b give that A has excess energy E_x, $R(E_b|E_x)$, where "R" is used to avoid confusion with the collisional transition probability. When this probability is known, application of detailed balance allows the reverse microcanonical rate coefficients to be found (see ref. [5]). The fragmentation distribution is specified as follows:

```
<reaction id="R2">
    . . .
  <me:FragmentDist xsi:type="me:Prior"/>
  <me:MCRCMethod xsi:type="me:MesmerILT">
    <me:preExponential units="cm3molecule-1s-1">2.0e-12</me:preExponential>
    <me:activationEnergy units="kJ/mol">0</me:activationEnergy>
    <me:nInfinity>0.0</me:nInfinity>
  </me:MCRCMethod>
</reaction>
```

At present, there are two functional forms for this distribution:

`me:Prior:` This form is identical to that described above and is the default form.

`me:modPrior:` This form is based on the prior distribution, but the density of states of the deficient species is modified by a power, that is:

$$R(E_b|E_x) = \frac{(N_B(E_b))^m [N_t \otimes N_C](E_x - E_b)}{[(N_B)^m \otimes N_t \otimes N_C](E_x)}$$ (11.19)

This model attempts to account for any non-RRKM effects where not all internal modes of the deficient species contribute to reaction (see, for example, Shannon et al. [25]). The m parameter can depend on temperature:

$$m(T) = m_0(T/T_r)^n \tag{11.20}$$

where, m_0 is a constant and T_r is a reference temperature. Thus, a full specification of this model that might be used in a fitting exercise is:

```xml
<reaction id="R3">
  <product>
    <molecule ref="C2" role="modelled"/>
  </product>
  <reactant>
    <molecule ref="HC(O)CO" role="modelled"/>
  </reactant>
  <reactant>
    <molecule ref="H2O" role="excessReactant"/>
  </reactant>
  <me:FragmentDist xsi:type="me:modPrior">
    <me:modPriorOrder lower="0.1" upper="1.0" stepsize="0.01"> 0.256437</me:modPriorOrder>
    <me:modPriorNexp lower="-1.0" upper="2.0" stepsize="0.01">1.49365</me:modPriorNexp>
    <me:modPriorNexp>0.0</me:modPriorNexp>
    <me:modPriorTref>298</me:modPriorTref>
  </me:FragmentDist>
  <me:excessReactantConc>1E14</me:excessReactantConc>
  <me:MCRCMethod xsi:type="me:MesmerILT">
    <me:preExponential units="cm3molecule-1s-1">2.0e-12</me:preExponential>
    <me:activationEnergy units="kJ/mol">0</me:activationEnergy>
    <me:nInfinity>0.0</me:nInfinity>
  </me:MCRCMethod>
</reaction>
```

12 Deprecated Keywords

The development of MESMER, like any code, often involves decisions that later need to be revised, either because the design is flawed, or unanticipated requirements emerge. In this section deprecated keywords are listed together with a replacement option and the last version of MESMER to support the keyword. The exact date when a given keyword will be withdrawn cannot be given but users should assume that it is will be unusable in the release subsequent to the specified last supported release.

SimpleRRKM

This keyword invoked the original RRKM method for calculating microcanonical rate coefficients, but this method has been extended to include other features, hence this keyword has been replaced.

Replacement: RRKM

Last Supported Release: MESMER 6.1

dictRef="me:deltaEDown"

This property was used to define the $\langle \Delta E \rangle_d$ parameter of the exponential down model of energy transfer, which was the only model available in earlier versions of MESMER. A typical use of this keyword would have been:

```
<property dictRef="me:deltaEDown">
  <scalar units="cm-1" >210.0</scalar>
</property>
<property dictRef="me:deltaEDownTExponent">
  <scalar referenceTemperature="298." >0.6</scalar>
</property>
```

Replacement:

Energy transfer models are now defined as plug in class (see section 11.2.2). The above specification can be replaced by the following energy transfer model specification:

```
<me:energyTransferModel xsi:type="me:ExponentialDown">
   <me:deltaEDown units="cm-1">210.0</me:deltaEDown>
   <me:deltaEDownTExponent referenceTemperature="298.">0.6</me:deltaEDownTExponent>
 </me:energyTransferModel>
```

Last Supported Release: MESMER 6.1

13 MESMER FAQs

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

This is discussed in section 9.1.1.1

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"/>
```

The exception to this is when Eckart tunnelling is considered. In this case the classical correction to the product energy is required and for this reason the vibrational frequencies of the product are also required as input.

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

MESMER has six 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

- BimolecularSinkReaction, which has two reactants, one excess and one deficient modelled molecule, and one product which is a sink molecule

- PseudoIsomerization, which has two reactants one execess and one deficient modelled molecule, and one product which is a modelled molecule.

See section 7.3.2 for more details.

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 7.

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).

Q. I get a warning about eigenvalue separation, what do I do?

At high temperatures the eigenvalues that govern chemical relaxation becomes similar in magnitude to those that govern energy transfer. The Bartis-Widom approach to the calculation of macroscopic rate coefficients depends on this separation of the two sets of eigenvalues, and so begins to fail at high temperatures. Under these conditions the description of macroscopic dynamics in terms of a simple law of mass action expressions is not, strictly speaking, possible. If possible the range of temperatures considered should exclude the higher temperatures (e.g. high temperature data should be omitted from a fitting exercise). Often this is not possible, (e.g. because the system in question is a combustion system) and under these circumstances it is best to examine the population profiles of the species involved. Care should to be taken with the interpretation of any time constants obtained with this latter approach.

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. More information is available in our publication on MESMER; Glowacki *et al*.[1]

For thorough reviews of the ME mathematical development in MESMER, we refer the readers to work by Pilling and Robertson[8, 26-27], Miller and Klippenstein[28-29] and Robertson.[30] Useful discussions of numerical precision issues may be found in Gannon *et al*. [31] and references therein, and some discussion of the standard ILT is found in Davies *et al*. [32]

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 [29]. They may only be used to solve the ME in the collisionless limit, or for a system that has a single isomer [28, 33]. Thus, the bulk of ME modelling for systems under conditions of relevance to atmospheric and combustion chemistry is restricted to a one-dimensional ME [29].

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, shown schematically in Figure 14:

$$\frac{d}{dt}n_i(E_i,t) = \omega \int_{E_{i,0}}^{\infty} P(E_i|E_i')\, n_i(E_i',t)\,dE_i' - \omega n_i(E_i,t) - \sum_{j \neq i}^{M} k_{ji}(E_i)\, n_i(E_i,t)$$

$$+ \sum_{j \neq i}^{M} k_{ij}(E_j)\, n_j(E_j,t) - k_{Pi}(E_i) n_i(E_i,t) - k_{Ri}(E_i) n_i(E_i,t) \qquad (14.1)$$

$$+ k_{Ri}(E_i) K_{Ri}^{eq} \frac{\rho_i(E_i) e^{-\beta E_i}}{Q_i(\beta)} n_R n_m$$

The first term in Eq. 14.1 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,[34] 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. $Q_i(\beta) = \sum_{E_i} \rho(E_i) e^{-\beta E_i}$, which is the rovibrational partition function for the molecular species corresponding to isomer i.
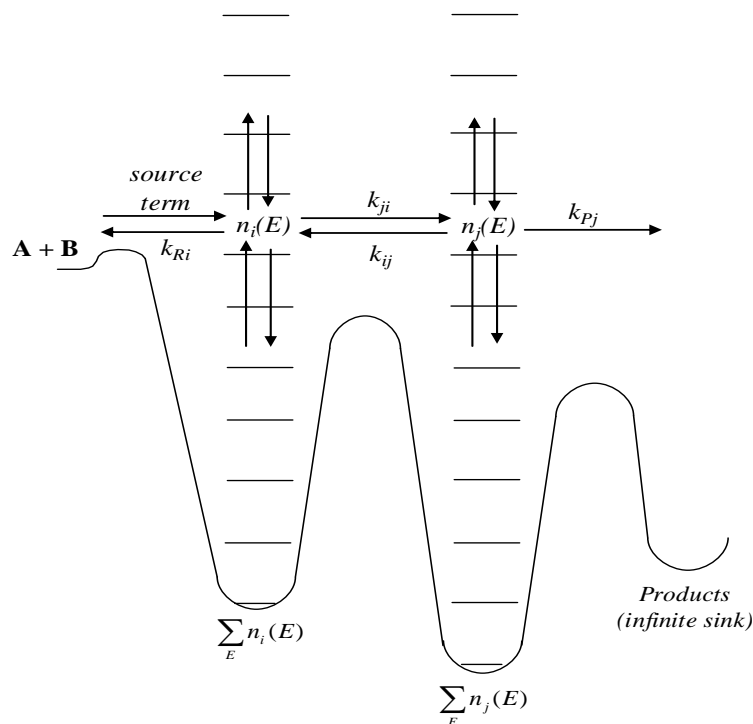
Figure 14: pictoral representation of Eq. 14.1 for a two well system, composed of isomers *i* and *j*. The reactants, A + B, are connected to isomer *i*, and the product channel is connected to isomer *j*. This figure shows the terms of Eq. 14.1 as they apply to one grain within each isomer, $n_i(E_i)$ 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:

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

(14.2)

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,[35] 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. 14.2 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).[32, 36] 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. 14.1.

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

$$P(E|E') = A(E')\exp\left(-(E'-E)/\langle\Delta E\rangle_d\right) \tag{14.3}$$

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. 14.3 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.[26] Other models with different transition probability distributions have been proposed,[26] such as Gaussian models[34] and double exponential models.[29] 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.[29] 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.[33] 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 1-D ME, this is manifest as an effective increase in $\langle \Delta E \rangle_d$ [33].

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 14.1 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 [26]. 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 [26, 29].

The coupled stochastic differential equations represented by Eq 14.1 may be reformulated as:

$$\frac{d}{dt}\boldsymbol{n} = \boldsymbol{M}\boldsymbol{n} \tag{14.4}$$

where $\boldsymbol{n}$ is a vector containing the populations of the grains for each isomer, $n_i(E_i, t)$, and $\boldsymbol{M}$ is the matrix that describes collisional energy transfer, as well as reactive loss and gain for each grain. In the case where Eq 14.1 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. 14.4 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:

$$\boldsymbol{n}(t) = \boldsymbol{U}e^{\boldsymbol{\Lambda}t}\boldsymbol{U}^{-1}\boldsymbol{n}(0) \tag{14.5}$$

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 [26, 28]. 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 inter-conversion between the different molecular configurations on the PES [28],[37]. 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) [8],[29].

The formulation of the EGME in terms of grains essentially corresponds to expanding the solutions of Eq. 14.4 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. 14.4, 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[28], 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 Miller *et al.*,[28] and one by Bartis and Widom [8],[38]. 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

$$A + B \rightleftharpoons C \qquad (14.6)$$

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

$$B \rightleftharpoons C \qquad (14.7)$$

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

$$k_a' x_B = k_d x_C \qquad (14.8)$$

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:

$$k_d = \sum_E k_d(E) f(E) \qquad (14.9)$$

where $f(E) = \rho(E) \exp(-\beta E) / Q_C$ and $Q_C = \sum_E f(E)$. Substituting Eq. 14.9 into 14.8 gives,

$$k_a' x_B = \sum_E k_d(E) f(E) x_C \tag{14.10}$$

where the term $f(E) x_C$ represents the equilibrium fraction in grain E of adduct C, and the $k_d(E)$ in 14.10 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, 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 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 S are related to the matrix elements of the asymmetric matrix M as follows:

$$S_{ij} = M_{ij} \left(\frac{f_j}{f_i} \right)^{1/2} = M_{ji} \left(\frac{f_i}{f_j} \right)^{1/2} = S_{ji} \tag{14.11}$$

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 S matrix elements for transition between grains in C and the bimolecular source term, B, are calculated using Eq. 14.11 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 14.10, and f_i is equivalent to x_B. Substituting these into Eq. 14.11, we obtain the matrix elements in S, which run from $E = 1$ to $E = N$:

$$S_{N+1,E} = S_{E,N+1} = k_d(E) \left(\frac{f(E) x_C}{x_B} \right)^{1/2} \tag{14.12}$$

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,

$$K = \frac{k_a'}{k_d} = \frac{x_C}{x_B} \tag{14.13}$$

then k_a' may be obtained by rearranging Eq. 14.13:

$$k'_a = Kk_d = S_{N+1,N+1} \tag{14.14}$$

14.1.2 Block Forms for Different Reaction Types

In this section, the forms of various reaction types are summarised in block form.

Simple dissociation:

$$\boldsymbol{M} = \omega_A(\boldsymbol{P_A} - \boldsymbol{I}) - \boldsymbol{K_{BA}}$$

where $\boldsymbol{M}$ is the collision, ω_A is the collision frequency, $\boldsymbol{P_A}$ is the transition probability matrix, $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{K_{BA}}$ and is a diagonal matrix containing the microcanonical rate coefficients of A going to B.

Isomerization:

$$\boldsymbol{M} = \begin{pmatrix} \omega_A(\boldsymbol{P_A} - \boldsymbol{I}) - \boldsymbol{K_{BA}} & \boldsymbol{K_{AB}} \\ \boldsymbol{K_{BA}} & \omega_B(\boldsymbol{P_B} - \boldsymbol{I}) - \boldsymbol{K_{AB}} \end{pmatrix}$$

Here there are additional terms for species B, but otherwise terms have the same meaning as for the simple dissociation case.

Association/Dissociation:

$$\begin{pmatrix} \omega_C(\boldsymbol{P_C} - \boldsymbol{I}) - \boldsymbol{K_C} & k_{f,\infty}[B]\boldsymbol{\varphi} \\ \boldsymbol{k_C} & -k_{f,\infty}[B] \end{pmatrix}$$

Here $\boldsymbol{k_C}$ is a row vector, $k_{f,\infty}$ is the high-pressure association rate coefficient, $[B]$ is the concentration of the excess species (assumed constant) and $\boldsymbol{\varphi}$ is the chemical activation distribution represented as a column vector.

Pseudoisomerization:

$$\begin{pmatrix} \omega_C(\boldsymbol{P_C} - \boldsymbol{I}) - \boldsymbol{K_C} & \boldsymbol{F_C}\boldsymbol{K_C^T}\boldsymbol{R^T}\boldsymbol{F_A^{-1}} \\ \boldsymbol{RK_C} & \omega_A(\boldsymbol{P_A} - \boldsymbol{I}) - \boldsymbol{K_A} \end{pmatrix}$$

Here the terms are similar to the isomerization case except for $\boldsymbol{R}$ which is the fragmentation distribution discussed in section 11.2.7, and $\boldsymbol{F_A}$ and $\boldsymbol{F_C}$ which are diagonal matrices of the Boltmann distributions of A and C respectively. The associated species is C and the upper right block shows the use of detailed balance to determine individual rate coefficients.

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. Hence

Eq. (14.7) fully describes the isomerization process, 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:

$$k_a x_B = k_d x_C \qquad (14.15)$$

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:

$$k_d = \sum_E k_d(E) f(E) \qquad (14.16)$$

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:

$$k_d(E) = \omega \sum_{i=E_0}^{i=E_t} P(i|E) \qquad (14.17)$$

where ω is the collision frequency. Substituting Eq. 14.16 into 14.15, we obtain:

$$k_a x_B = \sum_E k_d(E) f(E) x_C \qquad (14.18)$$

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. 14.17. 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, S, needs to have reservoir row and column vectors that are identical by detailed balance (see Eq. 14.11). 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. 14.11, we obtain the matrix elements in S for the reservoir state, which run from $E = 1$ to $E = N$:

$$S_{N+1,E} = S_{E,N+1} = k_d(E)\left(\frac{f(E)x_C}{x_B}\right)^{1/2} \tag{14.19}$$

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:

$$K = \frac{k_a}{k_d} = \frac{x_C}{x_B} \tag{14.20}$$

then k_a may be obtained by rearranging Eq. 14.20:

$$k_a = Kk_d = S_{N+1,N+1} \tag{14.21}$$

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.[39-40] Consider a two well isomerization system, after symmetrization the overall master equation matrix can be written as:

$$M = \begin{pmatrix} \omega(S_A - 1) & 0 \\ 0 & \omega(S_B - 1) \end{pmatrix} + \begin{pmatrix} -K_A & K' \\ K' & -K_B \end{pmatrix} \tag{14.22}$$

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 U_A and U_B, then the combined eigenvector matrix for the first term on the right-hand side can be written as:

$$U = \begin{pmatrix} U_A & 0 \\ 0 & U_B \end{pmatrix} \tag{14.23}$$

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

$$M' = UMU^T = \begin{pmatrix} \Lambda_A & 0 \\ 0 & \Lambda_B \end{pmatrix} + \begin{pmatrix} U_A & 0 \\ 0 & U_B \end{pmatrix} \begin{pmatrix} -K_A & K' \\ K' & -K_B \end{pmatrix} \begin{pmatrix} U_A^T & 0 \\ 0 & U_B^T \end{pmatrix} \tag{14.24}$$

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:

$$k(\beta) = \frac{1}{Q(\beta)} \int_0^\infty k(E)\rho(E) \exp(-\beta E) \, dE \tag{14.25}$$

Where $\rho(E)$ is the rovibrational density of states and $Q(\beta)$ is the corresponding canonical partition function. Substituting Eq. 14.2 for $k(E)$ gives,

$$k(\beta) = \frac{1}{hQ(\beta)} \int_0^\infty W(E) \exp(-\beta E) \, dE \tag{14.26}$$

and rearranging gives:

$$hQ(\beta)k(\beta) = \int_0^\infty W(E) \exp(-\beta E) \, dE = \mathcal{L}[W(E)] \tag{14.27}$$

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

$$W(E) = h\mathcal{L}^{-1}[Q(\beta)k(\beta)] \tag{14.28}$$

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

$$k(\beta) = A_0 \left(\frac{\beta_0}{\beta}\right)^n \exp\left(-\beta E_a\right) \tag{14.29}$$

it follows that:

$$\frac{W(E)}{h} = k(E)\rho(E) = A_0\beta_0^n \mathcal{L}^{-1}\left[\frac{Q(\beta)}{\beta^n}\exp\left(-\beta E_a\right)\right] \tag{14.30}$$

Further progress can be made by applying the convolution theorem:

$$\mathcal{L}^{-1}[Q(\beta)G(\beta)] = q \otimes g \tag{14.31}$$

where Q and q, and G and g are transform pairs, and $\otimes$ denotes convolution. Solution of 14.30 is possible by recognizing that

$$\mathcal{L}^{-1}[Q(\beta)] = \rho(E) \tag{14.32}$$

And

$$\mathcal{L}^{-1}[G(\beta)] = \mathcal{L}^{-1}\left[\frac{1}{\beta^n}\exp\left(-\beta E_a\right)\right] = \frac{(E - E_a)^{n-1}u(E - E_a)}{\Gamma(n)} \tag{14.33}$$

where $u(E - E_a) = 0$ if $E < E_a$ and is unity otherwise, is the Heavyside step function. Convolving the ILTs in Eq. 14.32 and 14.33 gives:

$$k(E)\rho(E) = \frac{A_0\beta_0^n}{\Gamma(n)}\int_0^E d\tau\rho(E - \tau)(\tau - E_a)^{n-1}u(\tau - E_a) \tag{14.34}$$

The units should be input follows: $\rho(E)$ in states cm^{-1}, A_0 in molecules cm^{-3} s^{-1}, and $\beta_0 = \frac{1}{k_B T_0}$ where k_B is in units of cm^{-1} K^{-1}, and T_0 in units of K. Note also that appearance of $\Gamma(n)$ in the denominator of Eq. 14.34 places a lower limit on the value of n, i.e. $n > 0$. For the case where $n = 0$, Eq. 14.34 simplifies to,

$$k(E)\rho(E) = A_0\rho(E - E_a) \tag{14.35}$$

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. 14.13:

$$k_d(\beta) = K_e(\beta)k_a(\beta) \tag{14.36}$$

If $k_a(\beta)$ has a modified Arrhenius form then the equivalent expression to Eq. 14.30 is,

$$\frac{W(E)}{h} = k(E)\rho(E) = A_0\beta_0^n \mathcal{L}^{-1}\left[K_e(\beta)\frac{Q(\beta)}{\beta^n}\exp\left(-\beta E_a\right)\right] \tag{14.37}$$

Solution of Eq. 14.37 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[32],

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

$$\tag{14.38}$$

where $\rho_R(E)$ is the convolved density of states for the associating pair, ΔH_0^0 is the enthalphy of reaction, μ is the reduced mass of the system and g_X is the spin degeneracy of species X. As with the unimolecular dissociation case, the appearance of $\Gamma(n+1.5)$ in the denominator of Eq. (14.38) places a limit on the value of n i.e. $n > -1.5$.

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,

$$Q_t = \left(\frac{2\pi mkT}{h^2}\right)^{3/2}V \tag{14.38}$$

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:

$$m = M/10^3 L \tag{14.39}$$

$$\beta' = hc/kT \tag{14.40}$$

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. 14.39 and 14.40 into 14.38 gives:

$$Q_t = \left(\frac{2\pi Mc}{10^3 L\beta' h}\right)^{3/2} V = C\left(\frac{M}{\beta'}\right)^{3/2} V \tag{14.41}$$

where C is given by

$$C = \left(\frac{2\pi c}{10^3 Lh}\right)^{3/2} \tag{14.42}$$

Volume is more conveniently expressed in cm^3 and to account for this Eq. 14.42 can be written as

$$Q_t = C'\left(\frac{M}{\beta'}\right)^{3/2} V' \tag{14.43}$$

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

$$C' = \left(\frac{2\pi c}{10^3 Lh}\right)^{3/2} 10^{-6} = \left(\frac{2\pi c}{10^7 Lh}\right)^{3/2} \tag{14.44}$$

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.

14.4 Classical Coupled Internal and External Rotational Densities of States

(Full details of the method used here are given by Gang et al [41]) For molecules with internal rotors, the rigid-rotor-harmonic-oscillator (RRHO) model is not appropriate, because the internal rotation couples to external rotation via Coriolis interactions. The situation is further complicated for molecules like butane and the higher alkanes, where the moments of inertia associated with external rotation depend upon the value of the generalized coordinates used to specify the internal rotation, that is there is an additional coupling term. Such molecules can be tackled by considering two sets of axes: a set of space fixed axes with corresponding coordinates s_i and a set of body fixed axes whose origin is located at the centre of mass of the

system and with corresponding coordinates r_i. The relation between the corresponding velocities is (see Goldstein or Wilson, Decius and Cross):

$$\dot{s}_\iota = \dot{r}_\iota + \omega \times r_i \tag{14.45}$$

where ω is the angular velocity. From this the usual (classical) kinetic energy expression can be obtained:

$$2T = m_i \dot{r}_i \cdot \dot{r}_i + 2\omega \cdot m_i r_i \times \dot{r}_i + \omega \cdot I \cdot \omega \tag{14.46}$$

where I is the usual definition of the (instantaneous) moment of inertia tensor. The coordinates r_i can be expressed in terms of internal coordinates, which might be the normal modes and/or the angles defining the internal rotation. As is usual, the middle (Coriolis) term of Eq. (14.46) will be assumed to be negligible for the high frequency-low amplitude modes and so they are decoupled from rotation and can treated separately. This is not true of internal rotation of course, but it does mean that we should be able to write a kinetic energy expression for this coupled system in the form:

$$2T = (\omega \quad \dot{\theta}) \begin{pmatrix} I & C^T \\ C & I_{int} \end{pmatrix} \begin{pmatrix} \omega \\ \dot{\theta} \end{pmatrix} = \dot{q}^T \cdot A(\theta) \cdot \dot{q} \tag{14.47}$$

where θ is a vector of internal rotor angle speeds, C are the Coriolis couplings from above and I_{int} are the inertial moments of motion about a given bond and the coupling between such motions. For later convenience, the kinetic energy quadratic form can be represented in more compact form as given by the last term in Eq. (14.47) where $\dot{q}^T = (\omega \quad \dot{\theta})$ and the matrix $A(\theta)$ represents the generalized inertia tensor and it a function of the internal rotational coordinates.

There are a number of ways that Eq. (14.47) can be used to obtain the density of states, here we begin by considering the classical canonical partition function for the combined internal and external rotation system,

$$Q(\beta) = \frac{1}{\sigma h^n} \int d\mathbf{p} \, d\mathbf{q} \, e^{-\beta H(\mathbf{p},\mathbf{q})} = \int_0^\infty \Omega(E) e^{-\beta E} \, dE \tag{14.48}$$

where n is the number of degrees of freedom (which in the present case is $3 +$ number of internal rotors), $H(\mathbf{p}, \mathbf{q})$ is the Hamiltonian if the combined internal and external rotation system, $\mathbf{q}$ are the generalized coordinates (the associated velocities of which appear in Eq. (14.47)) and $\mathbf{p}$ are the conjugate momenta, and σ accounts for symmetry. The Hamiltonian can be re-written as the sum of potential and kinetic energy terms giving:

$$Q(\beta) = \frac{1}{\sigma h^n} \int dq e^{-\beta U(q)} \int dp e^{-\beta p \cdot G \cdot p/2} \qquad (14.49)$$

where $U(q)$ is the potential energy associated with the internal rotations and the matrix G is related to the matrix A in Eq. (14.47) by the standard Legendre transform. The integral over the conjugate momenta can be done analytically, as can the integral over the Euler angles associated with external motion (assuming no external field), to give,

$$Q(\beta) = \frac{8\pi^2}{\sigma} \left(\frac{2\pi}{\beta h^2}\right)^{n/2} \int d\theta e^{-\beta U} \sqrt{|A(\theta)|} \qquad (14.50)$$

where $|A| (= |G|^{-1})$ is the determinant of matrix A. The standard thermodynamic functions can be obtained from this expression by the usual means and can be written as,

$$E = -\frac{1}{Q(\beta)} \frac{\partial Q(\beta)}{\partial \beta} = \frac{n}{2\beta} + \langle U \rangle \qquad (14.50)$$

$$C_V = -k\beta^2 \frac{\partial E}{\partial \beta} = \frac{nk}{2} + k\beta^2 (\langle U^2 \rangle - \langle U \rangle^2) \qquad (14.51)$$

where for an arbitrary function, f,

$$\langle f \rangle = \int d\theta f e^{-\beta U(\theta)} \sqrt{|A(\theta)|} / \int d\theta e^{-\beta U(\theta)} \sqrt{|A(\theta)|} \qquad (14.52)$$

To obtain the density of states for the combined internal and external rotation system the Laplace transform nature of Eq. (14.48) can be exploited, that is,

$$\Omega(E) = \mathcal{L}^{-1} \left[\frac{8\pi^2}{\sigma} \left(\frac{2\pi}{\beta h^2}\right)^{n/2} \int d\theta e^{-\beta U} \sqrt{|A(\theta)|} \right] \qquad (14.53)$$

It is useful to rearranged Eq. (14.53) to give,

$$\Omega(E) = \frac{8\pi^2}{\sigma} \left(\frac{2\pi}{h^2}\right)^{n/2} \mathcal{L}^{-1} \left[\left(\frac{1}{\beta}\right)^{\frac{n}{2}-1} \int d\theta \frac{e^{-\beta U}}{\beta} \sqrt{|A(\theta)|} \right] \qquad (14.54)$$

which, because it is the product of two functions, can be inverted to give a convolution,

$$\Omega(E) = \frac{8\pi^2}{\sigma} \left(\frac{2\pi}{h^2}\right)^{n/2} \frac{1}{\Gamma(\frac{n}{2}-1)} \int_0^E dx \, (E-x)^{\frac{n}{2}-2} \int d\theta u(x - U(\theta)) \sqrt{|A(\theta)|} \qquad (14.55)$$

where u is, again, the Heavyside step function. The inner integral (over $\boldsymbol{\theta}$) in evaluated in MESMER by Monte-Carlo integration using random $\boldsymbol{\theta}$ vectors generated from the Sobol sequence.

15 Revision History

15.1 MESMER 0.1 (Released 12/Jun/2009)

First beta release of MESMER.

15.2 MESMER 0.2 (Released 9/Jan/2011)

New features:

• Asymmetric rotor quantum mechanical energy levels.

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

• A plug-in class 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 added, allowing alternative calculation tasks to be implemented.

• A plug-in class 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.

15.3 MESMER 1.0 (Released 10/Feb/2012)

New features:

• 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 tunnelling method.

- Quantum mechanical hindered rotor calculations extended to include asymmetric potentials.

- A plug-in class implemented that calculates thermodynamic tables for species defined in the molecular list.

- A plug-in class added to allow the input of pre-defined coupled states.

15.4 MESMER 2.0 (Released 10/Feb/2013)

New features:

- Reaction types extended to include bimolecular sink reactions (See section 7.3.2).

- A plug-in class implemented that allows transition state sums of states to be input as a function of E or E and J (See section 11.2.4).

- A plug-in class implemented that calculates the Morse oscillator density of states (See section 11.2.3).

- A plug-in class implemented that calculates the prior distribution (See section 11.2.6).

- The ability to constrain parameters together during a fitting exercise (See section 8.5).
- Species evolution charts.

15.5 MESMER 3.0 (Released 24/Feb/2014)

New features:

- Extension of experimental specification to include bath gas.

- The ability to analyse Hessian matrices to exclude internal rotor modes prior to the determination of vibrational frequencies (See section 7.3.1(5) and 11.2.3).

- Addition of a new reaction type – pseudo isomerization (See section 7.3.2).

• Analytical representation of rate coefficients using Chebyshev polynomials (See section 11.2.1).

• XML Schema support to assist with input file specification (See section 7.2). Note the format of the MESMER file has changed between versions 2.0 and 3.0 in order that an XML schema can be written. However, MESMER 3.0 should be compatible with earlier version of the input files.

15.6 MESMER 4.0 (Released 16/May/2015)

New features:

• Internal rotor facility to, optionally, allow for an angle dependent internal moment of inertia term (See section 11.2.3).

• Multiple control blocks (See section 7.3.5).

• Limited support for second order bimolecular self-association reactions (See section 7.3.2).

• NASA polynomials (See section 11.2.1).

• Sensitivity analysis (See section 11.2.1).

• The accuracy of extended precision methods has been improved.

Bugs:

• An issue with the projection of internal rotor degrees of freedom from a Hessian has been addressed.

15.7 MESMER 4.1 (Released 27/Feb/2016)

New features:

• Thermodynamic table altered to allow both analytical and density of states based calculation of thermodynamic functions to be reported and so compared (see section 11.2.1).

• Heat capacity added to Thermodynamic table (see section 11.2.1).

• Error propagation method added (see section 11.2.1).

• Quantum mechanical hindered rotor method has been extended to include sine terms for the expansion of the angular dependent internal moment of inertia (see section 11.2.3).

Bugs:

• A bug has been fixed in the method for calculating the angular dependence on the internal moment of inertia. We thank Tanjin He, Tsinghua University, China, for assistance with this bug.

• A bug introduced in MESMER 4.0 that stopped a single source contributing to more than one well has been fixed (see HSO2.xml in the MesmerQA folder for an example). We thank Xiaoqing You, Tsinghua University, China, for assistance with this bug.

• A bug relating to the Analytical Representation feature, which led to an inverted temperature axis, has been rectified. We thank Hongmiao Wang, Tsinghua University, China, for assistance with this bug.

MESMER Input and Other Changes:

• The format of spin forbidden kinetics input has changed see section 11.2.4.

• The standard pressure used in the calculation of Gibbs energy and Entropy quoted in thermodynamic tables, has been altered from 1.01325×10^5 Pascals to 1 bar. We thank Branko Ruscic, Argonne National Laboratory, USA, for useful discussions on this point.

15.8 MESMER 5.0 (Released 7/Jun/2017)

New features:

• MPI parallelization of MESMER (see section 6.2).

• Extended fragmentation distribution (see section 11.2.7).

• New class for reading in density of states (see section 11.2.3).

• Implementation of the 1D Fourier grid vibrational state calculator (see section 11.2.3).

• Implementation of the Zhu-Nakamura non-adiabatic microcanonical rate coefficient algorithm (see section 11.2.4).

Other changes:

• The reporting of errors from a fitting exercise has been altered, such that if explicit error estimates for each data point appear in the input file then a goodness of fit is calculated, otherwise errors in parameters are estimated on the basis of χ^2 per degree of freedom of unity (see section 8.3.1).

• A new molecular property, `me:TSOpticalSymmetryNumber`, has been added that allows the specification of the number of transition state optical isomers (see section 7.3.1).

15.9 MESMER 5.1 (Released 25/Apr/2018)

New features:

• Analysis of trace data (see section 8.3.5).

• Addition of Canonical Rate Calculator for use in conjunction with irreversible exchange reactions (see section 11.2.4).

• Addition of full classical coupled internal-external rotor density of states calculator (see sections 11.2.3 and 14.4)

• Extension of the analytic representation facility to include the Plog representation (see section 11.2.1).

MESMER Input, bug fixes and Other Changes:

• An issue that caused MESMER 5.0 to crash on some Linux platforms when run in parallel has been addressed.

• An issue with the `derivedFrom` key word that allowed certain variables to be related during a fitting exercise has been addressed (see section 8.5).

• The keyword `RRKM` replaces the deprecated keyword `SimpleRRKM` for the specification of an RRKM based microcanonical rate calculator, though the latter keyword will continue to function.

• Further to user feedback, the keyword `me:ForceMacroDetailedBalance` has been set to apply at all times, that is detailed balance will be enforced when macroscopic rate coefficients are calculated. Previous behaviour can be restored by setting the element (but NOT its associated attribute!) `me:ForceMacroDetailedBalance` to `false` in the defaults.xml file (see section 7.3.5).

- The MesmerILT method for determining microcanonical rate coefficients has been altered so as to observe the limits on n parameter of the Arrhenius form $A_0 \left(\frac{\beta_0}{\beta}\right)^n \exp\left(-\beta E_a\right)$ (see section 14.3). We thank Timo Pekkanen for useful discussions regarding this parameter.

- The conversion factor used to convert inertia tensor elements into rotational constants has been altered so that it is consistent with the other fundamental constants used within MESMER. This may mean small changes in partition functions and densities of states.

For users compiling MESMER for parallel execution on Windows using Visual Studio, a new configuration "MPIRelease" has been added to simplify the procedure (see section 6.3.3).

15.10 MESMER 5.2 (Released 03/Jan/2019)

MESMER Input, bug fixes and Other Changes:

- An issue relating to electronic spin multiplicity of atomic species such that it was not being correctly accounted for when densities of states were convolved, has been addressed. We strongly recommend that users studying reactions involving atomic species re-run related calculations. We thank our colleague Prof. John Plane for bring this problem to our attention.

- An issue relating the convolution of the density of states, when the density of states has a very strong dependency on energy, has been addressed. We thank Charles McGill of North Carolina State University, US for bringing this problem to our attention.

- The control keyword `me:testMicroRates` has been altered to include attributes that allow the specification of the temperature range and interval over which the test should be performed (see section 7.3.5). We thank Charles McGill of North Carolina State University, US for suggesting this change.

- The set of energy units in which rotational constants can be specified has been extended to include GHz (see section 7.3.1). We thank Jesús Miguel Huamán Cjumo of Universidad Nacional de Ingeniería, Peru for suggesting this change.

• The output format for the analytical representation methods has been extended to include the NASA polynomial expressions for the stable species involved in the scheme. We thank Chong-Wen Zhou of Beihang University, Beijing, China for suggesting this change.

15.11   MESMER 6.0 (Released 9/May/2020)

New Features:

- The fitting of trace data has been extended to include trace weights ().
- The inverse Laplace transform (ILT) method has been extended to include an optionally second Arrhenius form (See section 11.2.4(2)).
- A new reaction type has been added that approximately models the distribution produced as a consequence of a bimolecular exchange reaction (See section 7.3.2).
- The Fourier Grid methods has been extended to allow the reduced (or effective) mass to be calculated for bend and wag modes (See section 11.2.3).
- The specification of the initial distribution of type Boltzmann has been extended to include an excitation energy to allow approximate modelling of photo-excitation (See section 11.2.6). We thank Emilio Martínez Núñez, Departamento de Química Física, Universidade de Santiago de Compostela, Spain for suggesting this change.
- Eckart Tunneling has been extended so that barrier heights (forward and reverse) can be specified explicitly. We thank Tom Stephenson, Swathmore College, U.S. for suggesting this change (See section 11.2.5).
- PTSets input has been extended to allow specification of intervals by a multiplicative factor, as well as an increment, allowing easier fall-off calculations (See section 7.3.3). We also thank Matt Johnson of MIT for bringing an issue with an earlier version of this facility to our attention.

MESMER Input, bug fixes and Other Changes:

- An issue with the basic potential used for the coupled classical rotors method has been rectified. We thank Timo Pekkanen of the University of Helsinki, for bringing the issue to our attention
- An issue relating to the calculation of equilibrium fractions for the case of very deep wells has been addressed. We thank our colleague Prof. John Plane for bring this problem to our attention.

148

- An issue with the processing of transition states by the Thermodynamic table method has been rectified. We thank Charles McGill of North Carolina State University for bring this issue to our attention.

15.12   MESMER 6.1 (Released 17/July/2021)

New Features:

- Biexponential down energy transfer model (see section 11.2.2).
- Lennard-Jones parameter calculation for compounds investigated by Jasper (see section 7.3.1).
- Extension of definition of diffusive loss in trace fitting (see section 8.3.5). We thank Timo Pekkanen of the University of Helsinki, for suggesting this change.

MESMER Input, bug fixes and Other Changes:

- QD libraries used to provided extended precision have been upgraded to version 2.3.22 (see section 6.4.2).
- Deprecate keywords, See section 12.
- An issue with reading the reference temperature used in the ILT method has been addressed.

We thank Timo Pekkanen and Prasenjit Seal of the University of Helsinki for bringing the following issues to our attention:

- An issue with extended ILT has been addressed.
- An issue with derived Arrhenius parameters, e.g., a given pre-exponential factor can now be derived from another pre-exponential factor, has been addressed.
- An issue with the weighted trace fitting for an equilibrating system with diffusive loss terms has been addressed.
- An issue with reactions with multiple transition states has been addressed.

16 References

1.   Glowacki, D. R.; Liang, C. H.; Morley, C.; Pilling, M. J.; Robertson, S. H. MESMER: An Open-Source Master Equation Solver for Multi-Energy Well Reactions. *J. Phys. Chem. A* **2012,** *116*, 9545-9560.

2.   Jasper, A. W. "Third-body" collision parameters for hydrocarbons, alcohols, and hydroperoxides and an effective internal rotor approach for estimating them. *Int. J. Chem. Kinet.* **2020,** *52*, 387-402.

3.   King, G. W.; Hainer, R. M.; Cross, P. C. The asymmetric rotor. I. Calculation and symmetry classification of energy levels. *J. Chem. Phys.* **1943,** *11*, 27-42.

4.   Miller, W. H. Tunneling Corrections to Unimolecular Rate Constants, with application to Formaldehyde. *J. Am. Chem. Soc.* **1979,** *101*, 6810-6814.

5.   Garrett, B. C.; Truhlar, D. G. Semi-Classical Tunneling Calculations. *J. Phys. Chem.* **1979,** *83*, 2921-2926.

6.   Green, N. J. B.; Robertson, S. H. General master equation formulation of a reversible dissociation/association reaction. *Chem. Phys. Lett.* **2014,** *605-606*, 44-46.

7.   Davis, M. J.; Klippenstein, S. J. Geometric investigation of association/dissociation kinetics with an application to the master equation for CH3+CH3 <-> C2H6. *J Phys Chem A* **2002,** *106*, 5860-5879.

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

9.   Blitz, M. A.; Green, N. J. B.; Shannon, R. J.; Pilling, M. J.; Seakins, P. W.; Western, C. M.; Robertson, S. H. Reanalysis of Rate Data for the Reaction CH3 + CH3 → C2H6 Using Revised Cross Sections and a Linearized Second-Order Master Equation. *J. Phys. Chem. A* **2015,** *119*, 7668-7682.

10.   McKee, K.; Blitz, M. A.; Pilling, M. J. Temperature and Pressure Studies of the Reactions of CH3O2, HO2, and 1,2-C4H9O2 with NO2. *J. Phys. Chem. A* **2015**, Ahead of Print.

11.   Tsang, W.; Bedanov, V.; Zachariah, M. R. Master Equation Analysis of Thermal Activation Reactions: Energy-Transfer Constraints on Falloff Behavior in the Decomposition of Reactive Intermediates with Low Thresholds. *J. Phys. Chem.* **1996,** *100*, 4011-18.

12.   Glowacki, D. R.; Rodgers, W. J.; Shannon, R.; Robertson, S. H.; Harvey, J. N. Reaction and relaxation at surface hotspots: using molecular dynamics and the energy-grained master equation to describe diamond etching. *Philosophical Transactions of the Royal Society a-Mathematical Physical and Engineering Sciences* **2017,** *375*.

13.   Seakins, P. W.; Robertson, S. H.; Pilling, M. J.; Slagle, I. R.; Gmurczyk, G. W.; Bencsura, A.; Gutman, D.; Tsang, W. Kinetics fo the Unimolecular Decomposition of i-C_3H_7 - Weak Collision Effects in Helium, Argon and Nitrogen. *J. Phys. Chem.* **1993,** *97*, 4450-4458.

14.   Brouard, M.; Macpherson, M. T.; Pilling, M. J. Experimental and RRKM Modelling Study of the CH_3 + H and CH_3 + D Reactions. *J. Phys. Chem.* **1989,** *93*, 4047-4059.

15.   Aubanel, E. E.; Wardlaw, D. M. Flexible Transition State Theory Rate Constants for the Recombination Reaction CH_3 + H = CH_4. *J. Phys. Chem.* **1989,** *93*, 3117-3124.

16.   Harvey, J. N.; Aschi, M. Modelling spin-forbidden reactions: recombination of carbon monoxide with iron tetracarbonyl. *Faraday Discuss.* **2003,** *124*, 129-143.

17. Zhu, C. Y.; Nakamura, H. THEORY OF NONADIABATIC TRANSITION FOR GENERAL 2-STATE CURVE CROSSING PROBLEMS .2. LANDAU-ZENER CASE. *J. Chem. Phys.* **1995,** *102*, 7448-7461.

18. Zhu, C. Y.; Nakamura, H. THEORY OF NONADIABATIC TRANSITION FOR GENERAL 2-STATE CURVE CROSSING PROBLEMS .1. NONADIABATIC TUNNELING CASE. *J. Chem. Phys.* **1994,** *101*, 10630-10647.

19. Naik, C.; Carstensen, H. H.; Dean, A. M. Reaction rate representation using Chebyshev polynomials. In *Spring Meeting of the Combustion Institute*, San Diego, CA, 2002.

20. Ziehn, T.; Tomlin, A. S. GUI-HDMR - A software tool for global sensitivity analysis of complex models. *Environmental Modelling and Software* **2009,** *24(7)*, 775-785.

21. Li, G. Y.; Wang, S. W.; Rabitz, H.; Wang, S. Y.; Jaffe, P. Global uncertainty assessments by high dimensional model representations (HDMR). *Chem Eng Sci* **2002,** *57*, 4445-4460.

22. Shannon, R. J.; Tomlin, A. S.; Robertson, S. H.; Blitz, M. A.; Pilling, M. J.; Seakins, P. W. Global Uncertainty Propagation and Sensitivity Analysis in the CH3OCH2 + O2 System: Combining Experiment and Theory To Constrain Key Rate Coefficients in DME Combustion. *J. Phys. Chem. A* **2015,** *119*, 7430-7438.

23. Sharma, S.; Raman, S.; Green, W. H. Intramolecular Hydrogen Migration in Alkylperoxy and Hydroperoxyalkylperoxy Radicals: Accurate Treatment of Hindered Rotors. *J. Phys. Chem. A* **2010,** *114*, 5689-5701.

24. Marston, C. C.; Balintkurti, G. G. THE FOURIER GRID HAMILTONIAN METHOD FOR BOUND-STATE EIGENVALUES AND EIGENFUNCTIONS. *J. Chem. Phys.* **1989,** *91*, 3571-3576.

25. Shannon, R. J.; Robertson, S. H.; Blitz, M. A.; Seakins, P. W. Bimolecular reactions of activated species: An analysis of problematic HC(O)C(O) chemistry. *Chem. Phys. Lett.* **2016,** *661*, 58-64.

26. Pilling, M. J.; Robertson, S. H. Master equation models for chemical reactions of importance in combustion. *Annu. Rev. Phys. Chem.* **2003,** *54*, 245-275.

27. Holbrook, K. A.; Pilling, M. J.; Robertson, S. H. *Unimolecular Reactions*. John Wiley & Sons: Chichester, 1996.

28. Klippenstein, S. J.; Miller, J. A. From the time-dependent, multiple-well master equation to phenomenological rate coefficients. *J. Phys. Chem. A* **2002,** *106*, 9267-9277.

29. Miller, J. A.; Klippenstein, S. J. Master equation methods in gas phase chemical kinetics. *J. Phys. Chem. A* **2006,** *110*, 10528-10544.

30. Robertson, S. H. Foundations of the Master Equation. In *Comprehensive Chemical Kinetics*, 2019; Vol. 43, pp 291-361.

31. Gannon, K. L.; Glowacki, D. R.; Blitz, M. A.; Hughes, K. J.; Pilling, M. J.; 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* **2007,** *111*, 6679-6692.

32. Davies, J. W.; Green, N. J. B.; Pilling, M. J. The Testing of Models for Unimolecular Decomposition via Inverse Laplace Transformation of Experimental Recombination Data. *Chem. Phys. Lett.* **1986,** *126*, 373-379.

33. Miller, J. A. Combustion chemistry: Elementary reactions to macroscopic processes - Concluding Remarks. *Faraday Discuss.* **2001,** *119*, 461-475.

34. Gilbert, R. G.; Smith, S. C. *Theory of Unimolecular and Recombination Reactions*. Blackwell Scientific Publications: Oxford, 1990.

35. Baer, T.; Hase, W. L. *Unimolecular Reaction Dynamics: Theory and Experiments*. Oxford University Press: New York, 1996.

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

37.   Green, N. J. B.; Bhatti, Z. A. Steady-state master equation methods. *Phys. Chem. Chem. Phys.* **2007,** *9*, 4275-4290.

38.   Bartis, J. T.; Widom, B. Stochastic Models of Interconversion of Three or More Chemical Species. *J. Chem. Phys.* **1974,** *60*, 3474-3482.

39.   Venkatesh, P. K.; Dean, A. M.; Cohen, M. H.; Carr, R. W. Master equation analysis of intermolecular energy transfer in multiple-well, multiple-channel unimolecular reactions .1. Basic theory. *J. Chem. Phys.* **1997,** *107*, 8904-8916.

40.   Venkatesh, P. K.; Dean, A. M.; Cohen, M. H.; 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. *J. Chem. Phys.* **1999,** *111*, 8313-8329.

41.   Gang, J.; Pilling, M. J.; Robertson, S. H. Monte Carlo calculation of partition functions for straight chain alkanes. *Chem. Phys.* **1998,** *231*, 183-192.