RMG Documentation

Release 4.0.1

W.H. Green et al.

May 06, 2013

CONTENTS

This user manual for RMG 4.0.1 was generated on May 06, 2013. For the latest version of this documentation, please visit http://rmg.sourceforge.net/

Please also refer to our website for how to cite RMG in published work.

CHAPTER

ONE

INTRODUCTION TO RMG

RMG is an automatic chemical reaction mechanism generator that constructs kinetic models composed of elementary chemical reaction steps using a general understanding of how molecules react. The model parameters (rate constants and reaction thermodynamics) are estimated using a database and the idea that the behavior of functional groups is somewhat independent of the molecule(s) containing them. The RMG database consists of two parts: kinetic rate rules and thermodynamic group additivity values.

RMG is an object-oriented program written in Java, which provides a stable, robust programming architecture that is easily extended, modified, and improved. At its core, RMG relies on two fundamental data structures: graphs and trees. The graphs represent the chemical structures, and the trees represent the databases of thermodynamic and kinetic data. Currently, RMG can generate reaction mechanisms for species involving carbon, hydrogen, and oxygen, and its mechanisms can contain many hundreds of species and tens of thousands of reactions. It is also capable of performing first-order sensitivity analysis on the rate constants and species thermodynamics.

RMG was originally developed by Dr. Jing Song under the guidance of Prof. William Green in the Department of Chemical Engineering at the Massachusetts Institute of Technology (http://web.mit.edu/). Currently there are several members of the Green group contributing to RMG's ongoing development. A full list of past and present RMG developers can be found on the *Credits* (page 151) page.

1.1 Changes

RMG 4.0.1 is a bug-fix release, primarily addressing a problem with liquid phase solvation energy predictions. Changes since RMG 4.0 include:

- Fix a bug predicting thermochemistry of species in solution at temperatures other than 298 K. (Prior versions had an incorrect enthalpy of solvation leading to erroneous temperature dependence, although the Gibbs free energy at 298 K would have been correct).
- Solvation parameter predictions improved for several classes of species.
- Example input files for using the stand-alone Abraham parameter estimator (for solvation predictions) are now provided.
- Seed mechanisms may now contain reactions with up to four products.

RMG 4.0 includes many new features and bugfixes. Changes since RMG 3.3 include:

- Support for sulfur (divalent) has been added, including many new thermo groups and reaction families.
- Improved support for automatic MM4 calculations to estimate the thermochemistry of cyclic molecules, including a new hybrid mode for reducing the number of MM4 calculations by first checking an existing database for corrections.
- A new reaction family for R group substitution on ethers has been added.
- RMG can now be constrained to only run pressure dependence for small molecules.
- Some Fortran modules now run as listener subprocesses to reduce the overhead of forking new processes, making RMG jobs run quite a bit faster.
- A new method for restarting jobs has been added, via saving of a new condition file on output with all of the final model core species.
- Some settings can now be optionally controlled using environment variables, particularly in the setting of output and scratch directories.
- Many new rate rules have been added, corrected, or otherwise improved.
- Many other bugfixes

RMG 3.3 includes a few new features and many bugfixes. Changes since 3.2.1 include the following:

Note: There have been several significant changes to the condition file syntax in RMG 3.3. Condition files created for earlier versions of RMG will not work with RMG 3.3. Please refer to *Creating a Condition File Manually* (page 31) for more information on the new syntax.

- Support for simulating *liquid-phase systems* (page 49), including solvation thermochemistry and diffusion-limited kinetics
- Correct negative activation energies predicted by kinetics estimation rules
- An option to specify multiple initial compositions (in addition to multiple temperatures and pressures) has been added
- Intermediate concentrations are *written to file* (page 83) during AUTO runs
- The model enlargement algorithm with pressure-dependence now uses species leak fluxes instead of network leak fluxes
- Pruning with pressure-dependence is now working
- Experimental support for *MM4 calculations for thermochemistry estimation* (page 36) (requires separate access to MM4 software)
- Cleaner, easier-to-read log information automatically and concurrently printed to the console and saved to file
- Several refinements and bugfixes to the pressure dependence functionality
- A number of various database updates and fixes

RMG 3.2.1 is a minor bug-fix release. Changes since 3.2 include the following:

- Better handling of pressure-dependent reaction network expansion when multiple reaction systems are used by restricting to one expansion per iteration. This could lead to "PDepException: Tried to determine nonincluded isomer with maximum leak flux, but there are no nonincluded reactions, so no isomer can be identified" errors.
- Pruning with pressure-dependence on is now enabled. Only species that have not been explored in any pressure-dependent reaction network are eligible for pruning.
- Only run pressure-dependence calculation when necessary: when one or more fully explored unimolecular isomer wells are present. This makes pressure-dependent mode run much faster.
- Many minor but important bugs fixed in pressure dependence module: subscript errors, etc.
- More detailed error messages when failing to read a reaction library, primary kinetic library, or seed mechanism.
- Some improved kinetics rate rules in the hydrogen abstraction reaction family.
- Clearer log messages when adding reactions from a Reaction Library
- Restart jobs should now be possible with Quantum Mechanics thermochemistry estimation

Changes in RMG 3.2 include the following:

- The `RMG_Database` folder has been restructured more logically. (*PrimaryReactionLibrary* (page 46) and *Seed Mechanism* (page 47) specifications in condition files will need updating)
- More accurate thermodynamic quantities from PM3 calculations for cyclic species based on explicit 3D molecule geometries (provided the user has access to GAUSSIAN03 or MOPAC2009)
- Transport property estimation
- New Reaction Library option for considering (but not forcing) reactions not automatically identified by RMG's reaction templates (The old Primary Reaction Library has been renamed to Primary Kinetic Library.)
- New reaction model pruning option to reduce number of edge species and ease memory limitations.
- New non-negativity option for DASSL simulations. This offers a new way of dealing with NegativeConcentrationException cases.
- New reaction family, 1,2-Birad_to_alkene, for interconversion of alkene (triplet) biradicals and singlet alkenes
- It is possible to specify *forbidden structures* (page 34) in the input file.
- Various error messages have been improved to give more helpful information to the user.
- `RMG_Dictionary.txt` will now be written at every iteration, rather than just the final iteration.
- CHEMKIN input files from each iteration are saved to separate folders
- RMG output file includes timestamp and code revision information.
- RMG stores `Restart` files. The reading-in of these files allows RMG to restore the conditions (species, reactions, pressure-dependent networks, etc.) of a converged mechanism without having to re-run the entire simulation from scratch.

- Treatment of molecular oxygen has changed significantly, including support for both triplet ground state and singlet; oxygen should now be specified in biradical form rather than the O=O form used previously; see *"Representing oxygen"* (page 107) for details
- Various improvements to pressure-dependent network enlargement algorithm
- Fixed writing of reactions with Evans-Polanyi parameters to CHEMKIN input file
- Refinements to values in the database based on literature review.
- No longer falls into infinite loop when the simulation results in an invalid model but RMG cannot identify a suitable action to take to enlarge the model.
- Reservoir state method for evaluating pressure-dependent rate coefficients has been vetted and is now the recommended method for publication-quality models. The modified strong collision method remains available for preliminary exploration of reaction models.
- Pressure-dependence algorithm now defaults to an appropriate grid of temperatures and pressures based on the chosen interpolation model (e.g. Gauss-Chebyshev grid for Chebyshev polynomial model).
- Fixed several bugs relating to enlarging of partial pressure-dependent networks.
- Addressed issues that could lead to trying to simulate an empty reaction model in pressure-dependent cases.
- Fixed IWORK/RWORK renaming on Linux platforms.
- Fixed bug producing unneccessary error message (when using AUTO option) suggesting that the ODE solver failed when an edge flux exceeds user-specified tolerances at t=0.
- Fixed DASPK issues, including variables.dat renaming for multiple reaction systems, ability to use constant concentration, reaction flux evaluation for Lindemann reactions, and sensitivity coefficient calculation.
- Improved robustness for frequency estimation code ("frankie").
- Changed ODE Solver input file writing to use BufferedWriter to ease memory usage issues when using AUTO with large reaction models.
- Fixed a bug in counting pressure-dependent edge reactions with the AUTO method.
- Fixed calculation of Cp(T=0) and Cp(T=Infinity) for monoatomic and linear species.
- Addressed collider issues, including an ArrayOutOfBoundsException associated with colliders and a collider identification issue.
- Addressed infinite recursion error in getToEndOfAxis that could occur with cyclic cumulenic species.
- Fixed syntax for helium thermochemistry in CHEMKIN input file written by RMG.
- Fixed bugs in counting of core and edge reactions caused by pressure-dependent reactions and irreversible reactions.
- Primary Kinetics Library now correctly identifies matches for molecules with multiple resonance forms

Changes made in RMG 3.1 include the following:

- *PrimaryReactionLibrary* (page 46) and *PrimaryThermoLibrary* (page 33) now both behave as reference libraries from which data are taken (in preference to group additivity estimates), when and if the data are needed.
- *Seed Mechanism* (page 47) allows the mechanism building to start from a seed mechanism, which is included in its entirety before the simulation starts. (This is how *PrimaryReactionLibrary* (page 46) behaved in previous releases.)
- New databases: GRIMech 3.0 and PrIMe-recommended thermodynamic values are included
- Support for Chemkin's P-Log format for *k(T,P) reporting* (page 39) (in addition to Chebyshev format).
- Added additional options for the input file, including:
  - User-specified limits for number of carbon / oxygen / radical per species
  - User-specified Chebyshev fitting options
- Changed many dependent Fortran codes to use standard input and output, rather than writing temporary files to disk.
- Reduced run-to-run variations by standardizing the order of averaging of values in the kinetics trees.

Bug Fixes:

- Corrected inconsistencies in edge flux evaluation for pressure-dependent reactions.
- Corrected an error in inert gas normalization for runs with multiple temperatures/pressures
- Fixed DASPK interface
- Fixed bugs in peroxide (ROOR) frequency estimation
- Fixed a bug in the gauche correction database for alkenes
- Fixed issues with duplicate reactions and with Chebyshev fitting in writing CHEMKIN input files
- More frequent garbage collection

Changes made in RMG 3.0 include the following:

New features:

- Pressure-dependent reaction network generation
- InChI generation
- Graphical user interface for input file generation
- Generation of reaction mechanisms for multiple reaction conditions (T,P)
- Automatic time stepping option
- Updated RMG Viewer and Editor (including database editing tools)

Functionality changes:

- On the first step of mechanism generation, only one species is added, rather than adding all edge species

- Thermodynamics estimates for non-cyclic species incorporate certain steric effects (1,5-interactions and gauche interactions)

Bug fixes:

- Fixed bug in symmetry number generation (previously could be underestimated in certain cases)
- Fixed differential equations to correctly treat cases where total number of moles changes

CHAPTER

TWO

LICENSE

RMG is intended to be an open source program, available to the general public free of charge. The primary RMG code is distributed under the terms of the MIT/X11 License (http://www.opensource.org/licenses/mit-license.php), reproduced below. Although many of RMG's dependencies are also available under compatible open source licenses, there remain a few dependencies with more restrictive licenses. **It is the user's responsibility to ensure these licenses have been obtained.**

```
Copyright (c) 2002-2013 William H. Green and the RMG Team

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is furnished to do
so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

The FAME module for pressure dependent calculations uses LAPACK (http://www.netlib.org/lapack/), which has the following license:

```
Copyright (c) 1992-2008 The University of Tennessee.  All rights reserved.

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

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

- Redistributions in binary form must reproduce the above copyright
```

```
  notice, this list of conditions and the following disclaimer listed
  in this license in the documentation and/or other materials
  provided with the distribution.

- Neither the name of the copyright holders 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 PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

The contents of source/cclib/ are modified portions of cclib 1.0 (http://cclib.sourceforge.net) and form a software library available under the terms of the LGPL:

```
                  GNU LESSER GENERAL PUBLIC LICENSE
                       Version 2.1, February 1999

 Copyright (C) 1991, 1999 Free Software Foundation, Inc.
 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL.  It also counts
as the successor of the GNU Library Public License, version 2, hence
the version number 2.1.]

                            Preamble

The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some
specially designated software packages--typically libraries--of the
Free Software Foundation and other authors who decide to use it.  You
can use it too, but we suggest you first think carefully about whether
this license or the ordinary General Public License is the better
strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use,
not price.  Our General Public Licenses are designed to make sure that
you have the freedom to distribute copies of free software (and charge
for this service if you wish); that you receive source code or can get
```

it if you want it; that you can change the software and use pieces of
it in new free programs; and that you are informed that you can do
these things.

To protect your rights, we need to make restrictions that forbid
distributors to deny you these rights or to ask you to surrender these
rights. These restrictions translate to certain responsibilities for
you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis
or for a fee, you must give the recipients all the rights that we gave
you. You must make sure that they, too, receive or can get the source
code. If you link other code with the library, you must provide
complete object files to the recipients, so that they can relink them
with the library after making changes to the library and recompiling
it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the
library, and (2) we offer you this license, which gives you legal
permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that
there is no warranty for the free library. Also, if the library is
modified by someone else and passed on, the recipients should know
that what they have is not the original version, so that the original
author's reputation will not be affected by problems that might be
introduced by others.

Finally, software patents pose a constant threat to the existence of
any free program. We wish to make sure that a company cannot
effectively restrict the users of a free program by obtaining a
restrictive license from a patent holder. Therefore, we insist that
any patent license obtained for a version of the library must be
consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the
ordinary GNU General Public License. This license, the GNU Lesser
General Public License, applies to certain designated libraries, and
is quite different from the ordinary General Public License. We use
this license for certain libraries in order to permit linking those
libraries into non-free programs.

When a program is linked with a library, whether statically or using
a shared library, the combination of the two is legally speaking a
combined work, a derivative of the original library. The ordinary
General Public License therefore permits such linking only if the
entire combination fits its criteria of freedom. The Lesser General
Public License permits more lax criteria for linking other code with
the library.

We call this license the "Lesser" General Public License because it
does Less to protect the user's freedom than the ordinary General
Public License. It also provides other free software developers Less

of an advantage over competing non-free programs. These disadvantages
are the reason we use the ordinary General Public License for many
libraries. However, the Lesser license provides advantages in certain
special circumstances.

For example, on rare occasions, there may be a special need to
encourage the widest possible use of a certain library, so that it becomes
a de-facto standard. To achieve this, non-free programs must be
allowed to use the library. A more frequent case is that a free
library does the same job as widely used non-free libraries. In this
case, there is little to gain by limiting the free library to free
software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free
programs enables a greater number of people to use a large body of
free software. For example, permission to use the GNU C Library in
non-free programs enables many more people to use the whole GNU
operating system, as well as its variant, the GNU/Linux operating
system.

Although the Lesser General Public License is Less protective of the
users' freedom, it does ensure that the user of a program that is
linked with the Library has the freedom and the wherewithal to run
that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and
modification follow. Pay close attention to the difference between a
"work based on the library" and a "work that uses the library". The
former contains code derived from the library, whereas the latter must
be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other
program which contains a notice placed by the copyright holder or
other authorized party saying it may be distributed under the terms of
this Lesser General Public License (also called "this License").
Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data
prepared so as to be conveniently linked with application programs
(which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work
which has been distributed under these terms. A "work based on the
Library" means either the Library or any derivative work under
copyright law: that is to say, a work containing the Library or a
portion of it, either verbatim or with modifications and/or translated
straightforwardly into another language. (Hereinafter, translation is
included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for

making modifications to it. For a library, complete source code means
all the source code for all modules it contains, plus any associated
interface definition files, plus the scripts used to control compilation
and installation of the library.

Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope. The act of
running a program using the Library is not restricted, and output from
such a program is covered only if its contents constitute a work based
on the Library (independent of the use of the Library in a tool for
writing it). Whether that is true depends on what the Library does
and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's
complete source code as you receive it, in any medium, provided that
you conspicuously and appropriately publish on each copy an
appropriate copyright notice and disclaimer of warranty; keep intact
all the notices that refer to this License and to the absence of any
warranty; and distribute a copy of this License along with the
Library.

You may charge a fee for the physical act of transferring a copy,
and you may at your option offer warranty protection in exchange for a
fee.

2. You may modify your copy or copies of the Library or any portion
of it, thus forming a work based on the Library, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) The modified work must itself be a software library.

    b) You must cause the files modified to carry prominent notices
    stating that you changed the files and the date of any change.

    c) You must cause the whole of the work to be licensed at no
    charge to all third parties under the terms of this License.

    d) If a facility in the modified Library refers to a function or a
    table of data to be supplied by an application program that uses
    the facility, other than as an argument passed when the facility
    is invoked, then you must make a good faith effort to ensure that,
    in the event an application does not supply such function or
    table, the facility still operates, and performs whatever part of
    its purpose remains meaningful.

    (For example, a function in a library to compute square roots has
    a purpose that is entirely well-defined independent of the
    application. Therefore, Subsection 2d requires that any
    application-supplied function or table used by this function must
    be optional: if the application does not supply it, the square
    root function must still compute square roots.)

These requirements apply to the modified work as a whole. If
identifiable sections of that work are not derived from the Library,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works. But when you
distribute the same sections as part of a whole which is a work based
on the Library, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote
it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Library.

In addition, mere aggregation of another work not based on the Library
with the Library (or with a work based on the Library) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public
License instead of this License to a given copy of the Library. To do
this, you must alter all the notices that refer to this License, so
that they refer to the ordinary GNU General Public License, version 2,
instead of to this License. (If a newer version than version 2 of the
ordinary GNU General Public License has appeared, then you can specify
that version instead if you wish.) Do not make any other change in
these notices.

Once this change is made in a given copy, it is irreversible for
that copy, so the ordinary GNU General Public License applies to all
subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of
the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or
derivative of it, under Section 2) in object code or executable form
under the terms of Sections 1 and 2 above provided that you accompany
it with the complete corresponding machine-readable source code, which
must be distributed under the terms of Sections 1 and 2 above on a
medium customarily used for software interchange.

If distribution of object code is made by offering access to copy
from a designated place, then offering equivalent access to copy the
source code from the same place satisfies the requirement to
distribute the source code, even though third parties are not
compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the
Library, but is designed to work with the Library by being compiled or
linked with it, is called a "work that uses the Library". Such a

work, in isolation, is not a derivative work of the Library, and
therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library
creates an executable that is a derivative of the Library (because it
contains portions of the Library), rather than a "work that uses the
library". The executable is therefore covered by this License.
Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file
that is part of the Library, the object code for the work may be a
derivative work of the Library even though the source code is not.
Whether this is true is especially significant if the work can be
linked without the Library, or if the work is itself a library. The
threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data
structure layouts and accessors, and small macros and small inline
functions (ten lines or less in length), then the use of the object
file is unrestricted, regardless of whether it is legally a derivative
work. (Executables containing this object code plus portions of the
Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may
distribute the object code for the work under the terms of Section 6.
Any executables containing that work also fall under Section 6,
whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or
link a "work that uses the Library" with the Library to produce a
work containing portions of the Library, and distribute that work
under terms of your choice, provided that the terms permit
modification of the work for the customer’s own use and reverse
engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the
Library is used in it and that the Library and its use are covered by
this License. You must supply a copy of this License. If the work
during execution displays copyright notices, you must include the
copyright notice for the Library among them, as well as a reference
directing the user to the copy of this License. Also, you must do one
of these things:

    a) Accompany the work with the complete corresponding
    machine-readable source code for the Library including whatever
    changes were used in the work (which must be distributed under
    Sections 1 and 2 above); and, if the work is an executable linked
    with the Library, with the complete machine-readable "work that
    uses the Library", as object code and/or source code, so that the
    user can modify the Library and then relink to produce a modified
    executable containing the modified Library. (It is understood
    that the user who changes the contents of definitions files in the
    Library will not necessarily be able to recompile the application

to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute
the Library except as expressly provided under this License. Any
attempt otherwise to copy, modify, sublicense, link with, or
distribute the Library is void, and will automatically terminate your
rights under this License. However, parties who have received copies,
or rights, from you under this License will not have their licenses
terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not
signed it. However, nothing else grants you permission to modify or
distribute the Library or its derivative works. These actions are
prohibited by law if you do not accept this License. Therefore, by
modifying or distributing the Library (or any work based on the
Library), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the
Library), the recipient automatically receives a license from the
original licensor to copy, distribute, link with or modify the Library
subject to these terms and conditions. You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties with
this License.

11. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License. If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Library at all. For example, if a patent
license would not permit royalty-free redistribution of the Library by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any
particular circumstance, the balance of the section is intended to apply,
and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system which is
implemented by public license practices. Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Library under this License may add
an explicit geographical distribution limitation excluding those countries,
so that distribution is permitted only in or among countries not thus
excluded. In such case, this License incorporates the limitation as if
written in the body of this License.

13. The Free Software Foundation may publish revised and/or new
versions of the Lesser General Public License from time to time.
Such new versions will be similar in spirit to the present version,
but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library
specifies a version number of this License which applies to it and
"any later version", you have the option of following the terms and
conditions either of that version or of any later version published by
the Free Software Foundation. If the Library does not specify a
license version number, you may choose any version ever published by
the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free
programs whose distribution conditions are incompatible with these,
write to the author to ask for permission. For software which is
copyrighted by the Free Software Foundation, write to the Free
Software Foundation; we sometimes make exceptions for this. Our
decision will be guided by the two goals of preserving the free status
of all derivatives of our free software and of promoting the sharing
and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO
WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR
OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY
KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME
THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU
FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A
FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF

SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the library's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This library is free software; you can redistribute it and/or
> modify it under the terms of the GNU Lesser General Public
> License as published by the Free Software Foundation; either
> version 2.1 of the License, or (at your option) any later version.
>
> This library is distributed in the hope that it will be useful,
> but WITHOUT ANY WARRANTY; without even the implied warranty of
> MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
> Lesser General Public License for more details.
>
> You should have received a copy of the GNU Lesser General Public
> License along with this library; if not, write to the Free Software
> Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the
> library 'Frob' (a library for tweaking knobs) written by James Random Hacker.
>
> <signature of Ty Coon>, 1 April 1990
> Ty Coon, President of Vice

That's all there is to it!

CHAPTER

THREE

GETTING AND INSTALLING RMG

3.1 License Requirements

RMG is intended to be an open source program, available to the general public free of charge. The primary RMG code is distributed under the terms of the MIT/X11 License (http://www.opensource.org/licenses/mit-license.php). Although many of RMG's dependencies are also available under compatible open source licenses, there remain a few dependencies with *more restrictive licences* (page 9). It is the user's responsibility to ensure these licenses have been obtained.

3.2 Installation instructions

3.2.1 Downloading RMG

Windows Installer Package

An installer is available for Windows users which provides RMG binaries and source code and automatically performs other installation tasks. This is the easiest method for RMG installation on Windows. Get the Windows installer at

---

Download

RMG 4.0.1 Installer for Windows (http://sourceforge.net/projects/rmg/files/rmg/RMG%204.0.1/rmg-4.0.1-win32-setup.exe/download) (32-bit only)

The installer provides a 32-bit version of RMG. You must compile RMG yourself to obtain 64-bit versions of the binaries.

For help with the installation on Windows, please refer to the *Installing on Windows* (page 22) section.

Source Code

There are three ways of obtaining the RMG source package. For all cases, installation instructions are available for *Windows* (page 22), *Linux* (page 24), and *Mac OS* (page 26).

Stable Release

A tarball of the most recent stable release can be downloaded here:

Download

RMG 4.0.1 Source Package (http://sourceforge.net/projects/rmg/files/rmg/RMG%204.0.1/rmg-4.0.1-source.tar.gz/download) (all platforms)

The above package is suitable for all platforms.

Latest Snapshot

The RMG development repository is publicly available on GitHub at

http://github.com/GreenGroup/RMG-Java/

To get the latest development version of the source code – in git parlance, the head of the official master branch – you can download a snapshot from

http://github.com/GreenGroup/RMG-Java/archives/master

Version Control

If you'd like to stay up-to-date with the latest development version as it evolves, you can use either git or subversion to checkout the latest version from our official repository. For git (recommended), use the command

```
$ git clone git://github.com/GreenGroup/RMG-Java.git
```

For subversion, use the command

```
$ svn checkout http://svn.github.com/GreenGroup/RMG-Java.git RMG-Java
```

This gives you read-only access. If you might want to make changes, a better idea is to create your own account on GitHub (http://github.com/), fork the official repository, and clone from your fork. More information on how to do this is available at http://help.github.com/.

3.2.2 Installing on Windows

This section is for users who wish to install RMG on computers running a Windows operating system. RMG is known to work on Windows XP, Windows Vista, and Windows 7. There are two methods of installing RMG on windows: an automatic installer (recommended) or a source package. You can find both of these on the Downloading RMG page.

Installer Package (Easiest Method)

1. Download the Windows installer file from the Downloading RMG page.
2. Run the installer file and follow the setup instructions. We recommend that you install to the default installation directory `C:\RMG` to (1) avoid spaces in the installation path and (2) avoid issues with administrative access, particularly in Windows Vista and Windows 7.

   The installer will install a complete set of binaries, source code, database, documentation, and examples by default. You can optionally choose not to install one or more of these things if you prefer. Only the binaries and database are required to run RMG.

   When the installation is complete, you are ready to use RMG.
3. To test that RMG is working, you can try to run one of the examples, such as the "minimal" example. The easiest way to do this is to double-click the `RMG.bat` file found in the folder containing the example.

Now that you have a working version of RMG, you may wish to install some of the *optional features* (page 28). Once you are ready to run RMG, continue to learning how to create RMG input files, either *by hand* (page 31) or via the *graphical user interface* (page 65).

Compiling from Source

If you wish to compile RMG yourself, you will need to install a number of dependencies, described below.

Java SE Development Kit (JDK) You can download the official implementation (http://www.oracle.com/technetwork/java/javase/downloads/index.html) from Oracle (previously Sun Microsystems). We recommend Java SE Version 6.

A Fortran compiler On Windows the easiest Fortran compiler to install is g95 (http://www.g95.org/). The gfortran (http://gcc.gnu.org/wiki/GFortran) compiler from the GNU Compiler Collection (installable via MinGW (http://www.mingw.org/)) is also known to work.

The Basic Linear Algebra Subprograms (BLAS) and Linear Algebra PACKage (LAPACK) The easiest way to do this is to download our generic 32-bit DLL files for BLAS and LAPACK from http://github.com/GreenGroup/RMG-Java/downloads. These should be sufficient for most users.

Apache Ant ant (http://ant.apache.org/) is used to generate Java executables.

The procedure is as follows:

1. Obtain the source package using one of the methods described on the Downloading RMG page.
2. Unpack and move the source package contents to your desired installation directory. We recommend avoiding paths containing spaces and paths to folders that Windows places administrative access restrictions on. Something simple like `C:\RMG` works best.
3. Set the `RMG` environment variable to the path of the RMG installation directory.
   - Go to `Start --> Control Panel --> System` (may need to switch to Classic Control Panel view). Click on the `Advanced` tab, then click on `Environment Variables`.
   - In the lower half of the `Environment Variables` window for System variables, click `New`.

- In the `New System Variable` window, in the `Variable` name field, type the environment variable name (i.e. `JAVA_HOME`). In the `Variable` value field, type the directory in which it was installed.
- Click OK twice to confirm the change.

If they aren't already set, you will need to also set the `JAVA_HOME` and `ANT_HOME` environment variables to the installation directories for Java and ant, respectively, as well as add the folders containing the Java and ant binaries to the `PATH` environment variable.

4. Place the BLAS and LAPACK DLL files (`blas.dll` and `lapack.dll`) in the `%RMG%\bin` directory. You may need to create this directory manually.
5. Compile the programs. The easiest way to do the latter is to execute the file `make.bat` in the `%RMG%` directory, either through the command prompt or simply by double-clicking it.

   This batch script does not compile the Java source; for this you must run the command `ant jar` at the command prompt. If the Java source is compiled successfully, you will see `BUILD SUCCESSFUL` at the end of the console output.

   You should now have the files `RMG.jar`, `fame.exe`, `frankie.exe`, `GATPFit.exe`, and `dasslAUTO.exe` in your `%RMG%\bin` directory. If these are present, then you have successfully compiled RMG.
6. To test that RMG is working, you can try to run one of the examples, such as the "minimal" example. The easiest way to do this is to double-click the `RMG.bat` file found in the folder containing the example.

Now that you have a working version of RMG, you may wish to install some of the *optional features* (page 28). Once you are ready to run RMG, continue to learning how to create RMG input files, either *by hand* (page 31) or via the *graphical user interface* (page 65).

RMG Viewer/Editor (RMGVE)

The RMG Viewer/Editor (RMGVE) is an accompanying software which can be used for viewing the database of RMG and also the models generated by RMG. The RMGVE is present along with RMG in the `%RMG%\RMGVE` directory and need not be downloaded separately. To run the RMGVE just go to the `%RMG%\RMGVE` directory and execute `RMGVE 20080101.bat` on the command line. The RMGVE directly reads the RMG database `%RMG%\database\RMG_database` which includes the thermo data and the kinetics data. Please refer to *Using the RMGVE* (page 87) to see how to modify the RMG database.

3.2.3 Installing on Linux

This section is for users who wish to install RMG on Linux and other Unix-like systems. RMG should successfully run on a variety of distributions, including Ubuntu (http://www.ubuntu.com/).

Dependencies

To install RMG on a Linux-based system, you must first install the following dependencies. In general, the first place you should go to install these dependencies is your distribution's software package managing

software, as suitable software packages are usually available via your distribution's online repository, if not already installed.

Java SE Development Kit (JDK) Version 6 of the official implementation (http://www.oracle.com/technetwork/java/javase/downloads/index.html) from Oracle (previously Sun Microsystems) is known to work; OpenJDK (http://openjdk.java.net/) may also work but has not been tested by the RMG team.

A Fortran compiler We recommend gfortran (http://gcc.gnu.org/wiki/GFortran) from the free GNU Compiler Collection. g95 (http://www.g95.org/) is another free compiler that is known to work, but is less likely to be available within your distribution's package repository.

The Basic Linear Algebra Subprograms (BLAS) and Linear Algebra PACKage (LAPACK) You may need to install the development version of these packages in order to obtain the correct type of libraries for RMG to link to. BLAS and LAPACK may already be installed, as they are used in many common applications.

GNU make The provided build scripts have the form of Makefiles that are executed by GNU make (http://www.gnu.org/software/make/).

Apache Ant ant (http://ant.apache.org/) is used to generate Java executables.

Compiling From Source

1. If you have not already done so, download RMG using one of the methods described on the *downloading RMG* (page 21) page.
2. Unpack the RMG source code (if necessary) and move it to the desired installation directory. Depending on where you choose to install RMG, you may need superuser privileges to do this.
3. Set the `RMG` environment variable to the path of your desired installation directory. As an example, let us say you placed the RMG package files at `/usr/local/rmg`. If you are using a bash shell, use the command

   ```
   $ export RMG=/usr/local/rmg
   ```

   If you are using a C shell, use the command,

   ```
   $ setenv RMG /usr/local/rmg
   ```

   To avoid typing this line each time, append it to your `~/.bashrc` (bash shell) or `~/.cshrc` (C shell) file. This will cause the `RMG` variable to be automatically set each time you initialize a terminal in the future.
4. From the root of the RMG installation directory, run `make` to compile RMG. This will compile all of the Fortran dependencies and the main Java executable.

   ```
   $ cd $RMG
   $ make
   ```

   Without any options, `make` will assume that you are using `gfortran` as your compiler. If you are using `g95` instead, use this version of the `make` command instead:

```
$ make F90=g95
```

You can also compile the Java code independently by running the following from the root RMG installation directory:

```
$ ant jar
```

If all of the above steps were completed successfully, then you should have a working version of RMG ready for use. You should see several executables and one JAR file appear in the `$RMG/bin` folder.

5. To test that RMG is working, you can try to run one of the examples, such as the "minimal" example, via commands similar to

```
$ cd $RMG/examples/RMG/minimal
$ java -jar $RMG/bin/RMG.jar condition.txt
```

Now that you have a working version of RMG, you may wish to install some of the *optional features* (page 28). Once you are ready to run RMG, continue to learning how to create RMG input files, either *by hand* (page 31) or via the *graphical user interface* (page 65).

3.2.4 Installing on MacOS X

This section is for users who wish to install RMG on MacOS X 10.6 (Snow Leopard). Other versions of MacOS X will be similar, but have not been tested.

Dependencies

To install RMG on a MacOS X system, you must first install the following

MacOS XCode Developer Tools These are included on the install DVD that came with your Mac, or you can download them from Apple at http://developer.apple.com/technologies/tools/ (after free registration). This includes all the build tools you will need, and versions of the BLAS and LAPACK optimized for high performance vector and numerical computing.

A Fortran compiler We recommend gfortran (http://gcc.gnu.org/wiki/GFortran) from the free GNU Compiler Collection. One nice way to get gfortran is to install the homebrew (http://mxcl.github.com/homebrew/) package manager, then then type `brew install gfortran` in a terminal. Alternatively, get the version from http://r.research.att.com/tools/ (download universal binary (http://r.research.att.com/gfortran-4.2.3.dmg)).

g95 (http://www.g95.org/) is another free compiler that is known to work, if you prefer.

Compiling From Source

Note: Quick start for impatient people:

```
$ git clone git://github.com/GreenGroup/RMG-Java.git
$ cd RMG-Java
$ git checkout master
$ make MACOS=true
$ make test
$ echo "export RMG=`pwd`" >> ~/.bashrc
```

More detailed instructions:

1. If you have not already done so, download the RMG source code using one of the methods described on the *downloading RMG* (page 21) page.

2. Unpack the RMG source code (if necessary) and move it to the desired installation directory, such as an `RMG-Java` directory inside your Home directory.

3. Set the `RMG` environment variable to the path of your desired installation directory. As an example, let us say you placed the RMG package files at `/Users/yourname/RMG-Java`. Open a Terminal (http://macapper.com/2007/03/08/the-terminal-an-introduction/) window and type the command

   ```
   $ export RMG=/Users/yourname/RMG-Java
   ```

   To avoid typing this line each time you open a Terminal window, append it to your `~/.bashrc` file, by typing:

   ```
   $ echo "export RMG=/Users/yourname/RMG-Java" >> ~/.bashrc
   ```

   This will cause the `RMG` variable to be automatically set each time you start a Terminal window in the future.

4. From the root of the RMG installation directory, run `make MACOS=true` to compile RMG. This will compile all of the Fortran dependencies and the main Java executable.

   ```
   $ cd $RMG
   $ make MACOS=true
   ```

   If you are using `g95` instead of `gfortran`, use this version of the `make` command instead:

   ```
   $ make F90=g95
   ```

   You can also compile the Java code independently by running the following from the root RMG installation directory:

   ```
   $ ant jar
   ```

   If all of the above steps were completed successfully, then you should have a working version of RMG ready for use. You should see several executables and one JAR file appear in the `$RMG/bin` folder.

5. To test that RMG is working, simply type:

   ```
   $ make test
   ```

   Or you can try to run one of the other examples, such as the "minimal" example, via commands similar to

```
$ cd $RMG/examples/RMG/minimal
$ java -jar $RMG/bin/RMG.jar condition.txt
```

Now that you have a working version of RMG, you may wish to install some of the *optional features* (page 28). Once you are ready to run RMG, continue to learning how to create RMG input files, either *by hand* (page 31) or via the *graphical user interface* (page 65).

3.2.5 Optional Advanced Features

RMG has a number of additional features available which enable advanced functionality but require additional installation steps. The following sections discuss the additional features available in RMG and the dependencies and installation procedures for each.

InChI Generation

RMG can be configured to work with the IUPAC International Chemical Identifier (InChI) format for representing molecules. When InChI support is enabled, RMG can *generate InChI strings* (page 36) for molecules and convert from InChI strings and/or .mol files to the adjacency list format. To enable InChI support you must download and install the following:

The IUPAC International Chemical Identifier InChI version 1

Version 1.02beta is required. This is available from http://www.iupac.org/inchi/download/index.html.

For Windows, in the 1.02beta download package, navigate to the `cInChI/vc6_project/Release` directory and copy the file:*cInChI-1.exe* executable to the `$RMG/bin` folder.

For Linux, after downloading and unpacking the InChI source code, change to the `InChI-1-API/cInChI/gcc_makefile` directory and execute the makefile. Copy the resulting executable `cInChI-1.exe` to the `$RMG/bin` folder.

```
$ cd InChI-1-API/cInChI/gcc_makefile
$ make
$ cp cInChI-1 $RMG/bin
```

You are now ready to generate InChIs in RMG.

Sensitivity Analysis

RMG can optionally perform sensitivity analysis on the reaction models it generates. Currently RMG uses DASPK 3.1 to perform sensitivity analysis. Analytical Jacobian and partial derivatives with respect to rates of reactions and Gibbs free energy of molecules are generated using the automated differentiator of DAEPACK.

1. Make sure the DAEPACK library file is in a location where the Fortran linker can find it. On Linux, you can either make a symbolic link of the library file in a directory in your library file search path, e.g. `/usr/lib`, or add the path containing the library to the `LD_LIBRARY_PATH` environment variable.

2. Compile DASPK using the command

```
$ cd $RMG
$ make daspk
```

You are now ready to run RMG with sensitivity analysis support.

3D-geometry-based Thermodynamic Property Calculation

RMG can optionally use quantum chemistry calculations to estimate the thermodynamic properties of the species it generates. This feature requires several additional dependencies, including *either* the Gaussian 03 or MOPAC 2009 quantum chemistry package. These dependencies are:

Python

Version 2.5, 2.6, or 2.7 of Python (http://www.python.org/) is recommended. On Windows you will probably need to manually add the directory containing the python binary to the `PATH` environment variable.

NumPy

Version 1.3.0 or later of NumPy (http://numpy.scipy.org/) is required.

RDKit

Version Q2 2009 or later of RDKit (http://rdkit.org/) is required. The typical installation process for RDKit should set the RDBASE environment variable and no further action on the part of the user should be required. On Windows you will also need to set environment variables as described here (http://code.google.com/p/rdkit/wiki/InstallingOnWindows).

OpenBabel

Version 2.2.0 or later of OpenBabel (http://openbabel.org/) is required. On Windows the OpenBabel installer should update the `PATH` environment variable automatically.

InChI

Version 1.02beta is required. Installation instructions are as given *here* (page 28). Don't forget to place the InChI executable in the `$RMG/bin` directory.

SYMMETRY

Version 1.15 or later of SYMMETRY (http://www.cobalt.chem.ucalgary.ca/ps/symmetry/) is required. The `SYMMETRY.EXE` executable should be placed in the `$RMG/bin` directory. A pre-compiled Windows binary executable is available via the `symmdos.zip` file found in the aforementioned web folder.

MOPAC 2009

Available at http://openmopac.net/downloads.html. MOPAC may be obtained freely for academic, not-for-profit use. You only need to install one of MOPAC 2009 or Gaussian 03, though having both can be advantageous for handling troublesome cases.

On Windows, use the download called `MOPAC2009 for DOS under any Windows`; the installer should set the `MOPAC_LICENSE` environment variable automatically.

On Linux, you will need to set the MOPAC_LICENSE variable manually

(e.g. `export MOPAC_LICENSE=/opt/mopac/` in .bashrc or before each time RMG is run with MOPAC calculations).

Gaussian 03

The Gaussian web site is http://www.gaussian.com/. On Windows you will likely need to manually update the `PATH` environment variable to include the Gaussian installation directory.

A modified version of cclib (http://cclib.sourceforge.net/) based on the cclib 1.0 release is also included with RMG; you do not need to download this separately.

CHAPTER

FOUR

CREATING A CONDITION FILE MANUALLY

To run RMG, you must specify the initial conditions and the model generation parameters, all of which are contained in a single file, `condition.txt`. The file `condition.txt` can be located in any directory, although it is advised that you create a special directory for RMG simulations, since each `condition.txt` creates its own subdirectories.

The `condition.txt` file should specify the following conditions, in order:

1. The main *Database* (page 32)
2. Any *restrictions* (page 32) on the number of atoms / radicals for all generated species
3. The name and location of all *Primary Thermodynamic Libraries* (page 33)
4. The name and location of all *Primary Transport Libraries* (page 33)
5. Optionally, some additional *forbidden structures* (page 34)
6. Whether *Restart* (page 34) files should be written or read
7. The *initial temperature and pressure* (page 35) of the system
8. Commands for *liquid phase* (page 49) systems
9. Whether to generate an *IUPAC International Chemical Identifier (InChI)* (page 36) for each species
10. (Optional) parameters for performing on-the-fly *quantum calculations* (page 36) as a substitute for group additivity
11. The name, concentration, and chemical structure of the *reactants* (page 37)
12. The specification and concentration of the *inert gas(es)* (page 38)
13. The method to use to estimate *spectroscopic data* (page 39) for each species
14. The *Pressure dependence* (page 39) model
15. The *finish controller* (page 41)
16. The *dynamic simulator* (page 42)
17. *Sensitivity analysis* (page 44) settings

18. The name and location of all *Primary Kinetic Libraries* (page 45)
19. The name and location of all *Primary Reaction Libraries* (page 46)
20. The name and location of all *Seed Mechanisms* (page 47)
21. The *units* (page 48) for the Arrhenius parameters A and Ea (to be reported in the generated chem.inp file)

The name `condition.txt` is variable. As you'll see in Section *Running RMG from the Command Line in Linux* (page 79), you can change the name of the initialization file, so long as it remains a `.txt` file.

4.1 Database

Field is **Required**

In the main RMG directory is a directory of databases: `$RMG/databases/`. By default, there is one directory within the directory of databases: `$RMG/databases/RMG_database`. It is possible, however, to create multiple databases. For example, you can copy the entire contents of the default database into a new database, `$RMG/databases/my_database`, and change some of the values. Unless you have created your own databases, the Database setting should be left at the default value:

```
Database: RMG_database
```

4.2 Species Restrictions

Field is **Optional**

By default, RMG will allow no more than 30 carbon atoms in any given species (other such default restrictions are: no more than 10 oxygen atoms, no more than 10 radicals, no more than 100 heavy-atoms, etc.). If you would like to change the default settings, the syntax is as follows

```
MaxCarbonNumberPerSpecies:      30
MaxOxygenNumberPerSpecies:      10
MaxRadicalNumberPerSpecies:     10
MaxSulfurNumberPerSpecies:      10
MaxSiliconNumberPerSpecies:     10
MaxHeavyAtomNumberPerSpecies: 100
MaxCycleNumberPerSpecies:       10
```

Each of these fields is optional, but if present must be in this order; the default values for all fields are listed above. A "Heavy Atom" is defined in RMG as a non-hydrogen atom. If you expect species with fused rings you may want to raise MaxCycleNumberPerSpecies.

Changing the default values of these fields, in particular, lowering the maximum value per species, may help you if your simulations are running out of memory. Also, changing the default settings of these fields could be beneficial if you are only concerned with the decomposition of the initial species.

4.3 Primary Thermo Library

Field is **Required**

By default, RMG will calculate the thermodynamic properties of the species from Benson additivity formulas. In general, the group-additivity results are suitably accurate. However, if you would like to override the default settings, you may specify the thermodynamic properties of species in the primaryThermoLibrary. When a species is specified in the primaryThermoLibrary, RMG will automatically use those thermodynamic properties instead of generating them from Benson's formulas. Multiple libraries may be created, if so desired. The order in which the primary thermo libraries are specified is important: If a species appears in multiple primary thermo libraries, the first instance will be used.

Please see Section *Editing the Thermodynamic Database* (page 87) for details on editing the primary thermo library. In general, it is best to leave the PrimaryThermoLibrary set to its default value. In particular, the thermodynamic properties for H and H2 must be specified in one of the primary thermo libraries as they cannot be estimated by Benson's method.

In addition to the default library, RMG comes with the thermodynamic properties of the species in the GRI-Mech 3.0 mechanism [GRIMech3.0] (page **??**).

This library is located in the `$RMG/databases/RMG_database/thermo_libraries` directory. All "Locations" for the PrimaryThermoLibrary field must be with respect to the `$RMG/databases/RMG_database/thermo_libraries` directory.:

```
PrimaryThermoLibrary:
Name: Default
Location: primaryThermoLibrary
Name: GRI-Mech 3.0
Location: GRI-Mech3.0
END
```

4.4 Primary Transport Library

Field is **Required**

By default, RMG will estiamte the transport properties (Lennard-Jones sigma and epsilon parameters) of a species from empirical correlations described by Tee et al. [Tee] (page 155); these correlations are functions of the acentric factor, the critical temperature, and the critical pressure. The acentric factor is estimated using the empirical correlation described by Lee et al. [Lee] (page 155), which itself is a function of the critical pressure, the critical temperature, and the boiling temperature. The critical properties and boiling point are estimated using the Joback group-additivity scheme [Joback] (page 155), [Reid] (page 155).

If you would like to override the default settings, you may specify the transport properties of species in the PrimaryTransportLibrary. Similar to the PrimaryThermoLibrary, RMG will automatically use those transport properties instead of estimating them using the methods described above. Multiple libraries may be created, if so desired. The order in which the primary transport libraries are specified is important: If a species appears in multiple primary transport libraries, the first instance will be used.

RMG comes with the transport properties of the species in the GRI-Mech 3.0 mechanism [GRIMech3.0] (page **??**). This library is located in the

`$RMG/databases/RMG_database/transport_libraries` directory. All "Locations" for the PrimaryTransportLibrary field must be with respect to the `$RMG/databases/RMG_database/transport_libraries` directory.

```
PrimaryTransportLibrary:
Name: GRI-Mech 3.0
Location: GRI-Mech3.0
END
```

4.5 Forbidden Structures

Field is **Optional**

It is possible, though not necessary, to specify additional forbidden structures in the condition file. This section should appear after the PrimaryTransportLibrary section. Each adjacency list can be a species, as described in the *Reactants* (page 37) section, but can contain *wildcards* (page 89) as described in the *Thermo Database and Adjacency List Notation* (page 88) section. For example:

```
ForbiddenStructures:
AdjacentBiradicalCs
1 C 1 {2,S}
2 C 1 {1,S}

AdjacentBiradicalCd
1 C 1 {2,D}
2 C 1 {1,D}

END
```

Note that each adjacency list must be followed by an empty line, and that the section must end (after an empty line) with the word *END*.

4.6 Restart Files

Field is **Required**

The next two items pertain to the reading and writing of "Restart" files.

For the baseball fans out there: Imagine a baseball game is in the top of the 5th inning, but the game is suspended (and later cancelled) due to bad weather. In baseball, the game would need to be re-played, starting from the first inning (unless it's the World Series, and then play will resume where it left off on the following day). RMG's Restart files fall under the latter condition: Imagine you have a terminated RMG simulation, and would like to pick up where you left off. Instead of having to re-run every aspect of the previous mechanism, you can supply RMG with the Restart files and RMG can more quickly restore the desired mechanism conditions.

The Restart files contain information regarding all species and reaction in the mechanism and are written after every successful ODE time step. These files are written in the `Restart` folder within the working directory. The files are:

```
coreSpecies.txt:   The name and connectivity of every species present in the model's core
coreReaction.txt:  The structure, kinetics, and comments for every high-P limit reaction pr
edgeSpecies.txt:   The name and connectivity of every species present in the model's edge
edgeReaction.txt:  The structure, kinetics, and comments for every high-P limit reaction pr
pdepreactions.txt: The structure, high-P limit kinetics, fall-off parameters, and comments
pdepnetworks.txt:  The structure, high-P limit kinetics, pressure-dependent kinetics, and c
```

These files may be used for a variety of reasons, including:

```
Having more detailed information on the model's edge species and reactions
As a Seed Mechanism, in constructing reaction mechanisms valid at multiple equivalence rati
As "Restart" files, in hopes of restoring the conditions of a terminated simulation
```

To write the files

```
ReadRestart: no
WriteRestart: yes
```

To read the files (Note: the `Restart` folder must be present in the working directory):

```
ReadRestart: yes
WriteRestart: no
```

4.7 Initial Conditions

Field is **Required**

The next item in the initialization file is the initial temperature and pressure. Currently, RMG can only model constant temperature and pressure systems. Future versions will allow for variable temperature and pressure. Please note that the temperature and pressure must be accompanied by a unit (in parentheses). Suitable temperature units are: K, F, and C. Suitable pressure units are: atm, torr, pa, and bar. The following example assumes that the system begins at 600 Kelvin and 200 atmospheres:

```
TemperatureModel: Constant (K) 600
PressureModel: Constant (atm) 200
```

You may also list multiple temperature and pressure conditions, as the example below shows:

```
TemperatureModel: Constant (K) 600 700 800 900
PressureModel: Constant (atm) 1 200
```

The intervals between the supplied values do not have to be equal, i.e. the following is an equally valid TemperatureModel field:

```
TemperatureModel: Constant (K) 600 700 900 1500
```

For cases in which multiple temperatures and pressures are listed, the model generator will simulate the system (still using an isothermal, isobaric model) at all possible combinations of the specified temperatures and pressures. Thus, the resulting reaction mechanism will be valid (within the given error tolerances, estimates, and various other assumptions) at all the conditions specified. (Note, however, that reported concentrations in the `Final_Model.txt` output file will only correspond to the first T/P combination.)

4.8 InChI Generation

Field is **Optional**

The next item in the initialization file is the InChI generation command. To use it you will need to *install the InChI software* (page 28). An InChI, or IUPAC **In**ternational **Ch**emical **I**dentifier, is an unique species identifier, with the exception of differing electronic or spin states, developed by IUPAC and NIST. An example of an InChI string (which represents the species caffeine) is:

```
InChI=1/C8H10N4O2/c1-10-4-9-6-5(10)7(13)12(3)8(14)11(6)2/h4H,1-3H3
```

The InChI is composed of layers, separated by a forward-slash (/):

1. Chemical formula layer
2. Connectivity layer
3. Hydrogen layer
4. ...

For the current release, the InChIs are used as another way to represent a molecule. In future releases, the InChIs will link RMG with an online thermochemical database: PrIMe, or **Pr**ocess **I**nformatics **M**ode**l**. The union will allow RMG to search for a community-recommended thermochemical data entry (e.g. Arrhenius parameters, NASA polynomials) before using estimation techniques.

If this field is turned on, the InChI for each species in the model is located in the `RMG_Dictionary.txt` file.

More information on the InChI project may be found at http://www.iupac.org/inchi/. More information on the PrIMe project may be found at http://primekinetics.org/

To enable the InChI generation, the item should read:

```
InChIGeneration: on
```

To disable the InChI generation, the item should read:

```
InChIGeneration: off
```

4.9 On-the-fly Quantum Calculations

Field is **Optional**

The next section is optional and specifies parameters for on-the-fly quantum mechanics (QM) calculations for estimating thermodynamic parameters. If the user does not have the needed *dependencies* (page 29) or wishes to use the group additivity approach for estimation of all unknown thermodynamic parameters, this section should be omitted. The first line in this block should be something like

```
ThermoMethod: QM gaussian03
```

The second field in the first line is optional and specifies the QM program to use. Options include “mopac”, “gaussian03”, or “both” (the default if this field is omitted). The “both” option requests that RMG first

try to use MOPAC and if all MOPAC attempts for a particular species are unsuccessful, then it will try Gaussian03. In all three cases, these will use PM3 calculations. Two experimental options are "mm4" and "mm4hr", and require the MM4 force field program; the former uses mm4 with the rigid-rotor, harmonic oscillator (RRHO) approximation, while the latter does rotor scans to treat rotor modes. It has been found that this approach generally works reasonably well, but currently fails to give accurate results for cyclic C3 species and may be less robust than the PM3 options.

The next line determines whether RMG will use QM calculations for all species or just cyclic ones:

```
QMForCyclicsOnly: on
```

It is often desireable to run with this option on, as the QM calculations can be time-consuming and may be less accurate than group-additivity for acyclic species, whereas they are more likely to improve accuracy for cyclic species.

The user must next specify the maximum radical number for species that will be fed to the QM program:

```
MaxRadNumForQM: 0
```

For molecules with more radicals than the value specified here, QM calculations will be performed on the saturated molecule (with added hydrogens) and hydrogen bond increment (HBI) corrections will be applied, similar to the approach used with group additivity. It is recommended that this value be left at 0 (zero), as it has been found that radicals tend to be difficult for the PM3 implementations to handle robustly, and errors or significant inaccuracies (particularly for biradicals) are more likely.

Next, the user must specify whether (and how) connectivity-checking functionality should be use:

    ConnectivityChecking: confirm

Options include "off", "check", and "confirm". The "off" option disables this functionality. The "check" option will perform a check on the final optimized geometry to try to assess whether it corresponds to a molecular structure with the desired connectivity; if not, a warning is printed, alerting the user. The "confirm" option will perform the same checks, except in cases where there is an apparent mismatch, the job will be treated as a failure and calculation with different geometry/keywords will be attempted.

Finally, you can choose whether or not to save the QM calculation input and output files. The benefits of saving them are that if another job encounters the same species it can re-use the result, if you have set them up to share the same RMG_QM_CALCS directory, and you can check the outputs if debugging. However, they can just take a lot of disk space, so if you'd rather not save them, then set it to 'no':

```
KeepQMFiles: no
```

4.10 Reactants

Field is **Required**

The name, concentration, and structure of each reactant must be specified. For each reactant, the first line should include the reactant's name (e.g. `C2O2`) and it's concentration(s) preceded by units (e.g. `(mol/cm3) 4.09e-3`). Acceptable units for concentration are "mol/cm3", "mol/m3", and "mol/l". Please note, RMG will **not** accept a species name that begins with a number. Be warned that names longer

than about 6 characters will be replaced with "SPC(#)" in the Chemkin file due to the maximum length of a reaction string in Chemkin, so keep your names short!

Like with the initial conditions, it is possible to set multiple starting concentrations, separated by spaces. RMG will create reaction systems for each combination of temperature, pressure, and starting concentration. You must enter the same number of starting concentrations for each species, including the *Inert Gases* (page 38) described in the next section.

The next line defines the reactant structure, described by an adjacency list. The *RMG Viewer and Editor* (page 89) is a useful tool for generating adjacency lists. Please note that you may choose the simplified adjacency list in which the hydrogen atoms are omitted (shown below):

```
InitialStatus:

CH3OH (mol/cm3)    4.09e-3
1  C 0 {2,S}
2  O 0 {1,S}

O2  (mol/cm3)      1.1E-7
1  O 1 {2,S}
2  O 1 {1,S}

END
```

Please note that the keyword `END` must be placed at the end of the InitialStatus section.

If one of the reactants is a resonance isomer, you only need to define one of the resonance structures, and RMG will automatically detect the others.

Represent molecular oxygen, O_2, **as shown above.** This is a deviation from versions of the RMG database prior to RMG 3.2. For more detail, please see this discussion on *representing oxygen* (page 107).

4.10.1 Maintaining a species at constant concentration

Adding the text `ConstantConcentration` after a species's initial concentration(s) in the condition file ensures that the concentration of the species remains unchanged during the simulation. This can be used for generating kinetic models in liquid phase experiments where gases (e.g. oxygen) are bubbled through the substrate at constant pressure, so that any consumption due to reactions is replenished externally:

```
O2 (mol/cm3) 4.09e-3  ConstantConcentration
1  O 1 {2,S}
2  O 1 {1,S}
```

4.11 Inert Gases

Field is **Required**

Following InitialStatus, the initialization file specifies which inert gases, if any, are used. Currently RMG can handle four inert gases: N_2, Ne, He, and Ar. If one of the gases is not used, set the concentration to `(mol/cm3) 0.0`. If no bath gas is used, set all concentrations to zero. In the example below, there is no nitrogen, the neon and helium are omitted, and the argon concentration is 2.21E-6 (mol/cm3):

```
InertGas:
N2   (mol/cm3)      0.0
Ar   (mol/cm3)     2.21E-6
END
```

In this case, the only difference between N2 and Ne (or He) is that N2 will be written to the SPECIES section of the `chem.inp` file, whereas Ne (or He) will be excluded.

To create valid CHEMKIN files, ensure that any 3rd-body collider molecules used in your Seed Mechanisms or Reaction Libraries are either declared in their species.txt files (if they are reactive) or in this section. (i.e. it's a good idea to list all of N2, Ne, He and Ar, even if their concentrations are 0)

The number of concentrations must correspond to the number of concentrations specified in the preceding *Reactants* (page 37) section.

Please note that the keyword `END` must be placed at the end of the InertGas section.

4.12 Spectroscopic Data

Field is **Required**

The next item in the initialization file is the method to use to estimate the spectroscopic data (i.e. rovibrational modes) of each species. At present there is only one method available, which utilizes approximate frequencies based on functional groups in the molecule when possible, then fits remaining degrees of freedom to heat capacity data. This can also be set to off if spectroscopic data is not needed (i.e. when pressure dependence is off).

When pressure dependent calculations are not desired, this item should read:

```
SpectroscopicDataEstimator: off
```

When pressure dependent calculations are desired, this item should read:

```
SpectroscopicDataEstimator: FrequencyGroups
```

4.13 Pressure Dependence

Field is **Required**

This flag is used to specify whether the model should account for pressure dependent rate coefficients. If pressure dependence is not desired, this item should read:

```
PressureDependence: off
```

If pressure dependence is desired, this item can take one of two values, based on the method used to approximate the pressure-dependent kinetics: `ModifiedStrongCollision` and `ReservoirState`. The former utilizes the modified strong collision approach of Chang, Bozzelli, and Dean [Chang2000] (page 155), and works reasonably well while running more rapidly. The latter utilizes the steady-state/reservoir-state approach of Green and Bhatti [Green2007] (page 155), and is more theoretically sound but more expensive.

To specify the modified strong collision approach, this item should read:

```
PressureDependence: ModifiedStrongCollision
```

To specify the reservoir state approach, this item should read:

```
PressureDependence: ReservoirState
```

For more information on the two methods, consult the following resources:

When pressure dependence is on, you must also specify a linear interpolation model to use to evaluate $k(T, P)$ and output to the Chemkin file. (This line is not required if pressure dependence is set to off.) This line must immediately follow the previous one.

By default pressure dependence is run for every system that might show pressure dependence, i.e. every isomerization, dissociation, and association reaction. In reality, larger molecules are less likely to exhibit pressure-dependent behavior than smaller molecules due to the presence of more modes for randomization of the internal energy. In certain cases involving very large molecules, it makes sense to only consider pressure dependence for molecules smaller than some user-defined number of atoms. This is specified e.g. using the line

```
MaxAtomsForPressureDependence: 20
```

to turn off pressure dependence for all molecules larger than the given number of atoms (20 in the above example). If this option is present, it must be given directly before the `PDepKineticsModel` option discussed below.

To disregard all temperature and pressure dependence and simply output the rate at the provided temperature and pressure, use the line

```
PDepKineticsModel: Rate
```

To use logarithmic interpolation of pressure and Arrhenius interpolation for temperature, use the line

```
PDepKineticsModel: PDepArrhenius
```

The auxillary information printed to the Chemkin chem.inp file will have the "PLOG" format. Refer to Section 3.5.3 of the `CHEMKIN_Input.pdf` document and/or Section 3.6.3 of the `CHEMKIN_Theory.pdf` document. These files are part of the CHEMKIN manual.

To fit a set of Chebyshev polynomials on inverse temperature and logarithmic pressure axes mapped to [-1,1], use the line

```
PDepKineticsModel: Chebyshev
```

You should also specify the number of temperature and pressure basis functions by adding the appropriate integers. For example, the following specifies that five basis functions in temperature and four in pressure should be used

```
PDepKineticsModel: Chebyshev 5 4
```

The auxillary information printed to the Chemkin chem.inp file will have the "CHEB" format. Refer to Section 3.5.3 of the `CHEMKIN_Input.pdf` document and/or Section 3.6.4 of the `CHEMKIN_Theory.pdf` document.

To generate the $k(T, P)$ interpolation model, a set of temperatures and pressures must be used. RMG can do this automatically, but it must be told a few parameters. Following the previous line, you should provide a line to specify the minimum and maximum temperature and number of temperatures to use in the grid, e.g.

```
TRange: (K) 300.0 2000.0 8
```

A similar line should follow for pressures:

```
PRange: (bar) 0.01 100.0 5
```

If you prefer, you can also specify the grid of temperatures and pressures manually using lines similar to the following:

```
Temperatures: 8 (K) 300.0 400.0 500.0 600.0 800.0 1000.0 1500.0 2000.0
Pressures: 5 (bar) 0.01 0.1 1.0 10.0 100.0
```

You should specify either a `TRange` or `Temperatures` line and either a `PRange` or `Pressures` line.

Note: If requesting "Chebyshev" polynomials in the "PDepKineticsModel" field, please consider the temperature(s) and pressure(s) that will be simulated in CHEMKIN (or similar). For example, if one were modeling a shock tube experiment, the largest temperature specified in the "Temperatures" or "TRange" field should probably be greater than 2000 K, as the simulated temperature will likely exceed this value. In the developers' experience, CHEMKIN will terminate a simulation when the simulated temperature exceeds the largest allowed temperature for the Chebyshev polynomials (the values of all system variables will be returned as "NaN").

Lastly, you also need to specify information regarding the number of energy grains to use when solving the Master Equation. The default value is 250; this was selected to balance the speed and accuracy of the Master Equation solver method. However, for some pressure-dependent networks, this number of energy grains will result in the pressure-dependent $k(T, P)$ being greater than the high-P limit. The default option in RMG is to print a warning message in the output file, stating this phenomenom, and then to continue the simulation. If you would prefer RMG to increase the number of energy grains and resolve the Master Equation, add the following line

```
DecreaseGrainSize: yes
```

If this field is turned on, RMG will increase the number of energy grains for the given pressure-dependent network until the computed $k(T, P)$ is less than the high-P limit or until the number of energy grains exceeds 1000. Please note, the runtime of a RMG simulation with this field turned on will be longer than one with this field turned off. Furthermore, not specifying this field in the input file is the same as

```
DecreaseGrainSize: no
```

4.14 Finish Controller

Field is **Required**

The Finish Controller defines the termination rules for model growth. Additionally, it includes the precision parameters such as reaction time, conversion, and error tolerance. The finish controller item should begin

with a line that contains only `FinishController:`.

Below the Finish Controller line, you must specify the goal. The goal tells the program when to terminate the reaction model expansion. There are two goal options: `ReactionTime` and `Conversion`. `ReactionTime` must be followed by a number and a time unit. Acceptable units of time are "sec", "min", "hr", and "day". `Conversion` must be followed by a reactant species name and a number between 0 and 1.

Beneath the goal, you must specify the error tolerance. Typical values for the error tolerance are between 0.0001 and 0.5. A smaller error tolerance generally corresponds to a larger model and longer model generation times. In the first example, the simulation will run for 0.001 seconds:

```
FinishController:
(1) Goal ReactionTime: 0.001 (sec)
(2) Error Tolerance: 0.1
```

In the second example, the simulation will run until 90% of the C_2H_6 reactant is consumed and has a much tighter error tolerance:

```
FinishController:
(1) Goal Conversion: C2H6 0.90
(2) Error Tolerance: 0.0001
```

For the initial model run, it is often preferable to use `Goal Conversion` instead of `Goal ReactionTime`, since it is difficult to judge *a priori* what the reaction time should be.

For a more detailed description on rate-based model enlargement, please consult [Susnow1997] (page 155).

4.15 Dynamic Simulator

Field is **Required**

RMG uses the DASSL or DASPK dynamic solver. DASSL is recommended, as RMG uses DASPK in a way that requires the use of the proprietary DAEPACK library, which is not distributed with RMG (see *installation instructions* (page 28) for details).

The first line beneath the solver should be either `TimeStep` or `Conversions`, depending on whether you chose Goal Conversion or Goal ReactionTime as your finish controller criterion.

The `TimeStep` indicates at what time steps RMG should check to see if the model needs to be enlarged. By default the units of time steps is **always** seconds. For example in the example shown below RMG will stop at 0.0001 sec, 0.0002 sec ... and so on to see whether the error tolerance is satisfied:

```
DynamicSimulator: DASSL
TimeStep: 0.0001 0.0002 0.0004 0.0006 0.0008 0.0009
```

The other option is to specify the Conversions. In this option the conversion of the FinishController species (in this case, C_2H_6) is monitored and RMG will stop at conversions of 0.1, 0.2, ... and so on to see whether the error tolerance is satisfied:

```
DynamicSimulator: DASSL
Conversions: 0.1 0.2 0.4 0.6 0.8 0.9
```

For both `TimeStep` or `Conversions`, in addition to determining when RMG will stop to check whether error tolerances are satisfied, the concentrations and flux at the points specified will be reported in the `Final_Model.txt` file.

Note that if conversion or time steps are specified this way, RMG only checks for model validity at those points and not in between. It is therefore advisable to use the automatic stepping option described next.

If you would not like to specify time steps or conversions, you may have them determined using the "automated time stepping" feature by specifying `AUTO` as follows:

```
DynamicSimulator: DASSL
TimeStep: AUTO
```

or:

```
DynamicSimulator: DASSL
Conversions: AUTO
```

The automated time stepping feature is described in greater detail in the Theory and Implementation Details *appendix* (page 136).

4.15.1 Pruning

When using automated time stepping, it is also possible to perform mechanism generation with pruning of "unimportant" edge species to reduce memory usage. This option is requested by specifying `AUTOPRUNE` in place of `AUTO`. When using `AUTOPRUNE`, this must be followed by four lines specifying pruning parameters, as the example below shows:

```
DynamicSimulator: DASSL
TimeStep: AUTOPRUNE
TerminationTolerance: 1.0E4
PruningTolerance: 1.0E-15
MinSpeciesForPruning: 1000
MaxEdgeSpeciesAfterPruning: 1000
```

`TerminationTolerance` and `PruningTolerance` are compared with the same quantity as `Error Tolerance` in the `FinishController` section (i.e. they are all compared with ratios of edge species flux to characteristic reaction model flux). `TerminationTolerance` indicates how high the edge flux ratio must get to interrupt the simulation (before reaching the `Goal Conversion` or `Goal ReactionTime` described earlier). Pruning won't occur if the simulation is interrupted before reaching the goal criteria, so set this high to increase pruning opportunities. The value should be greater than or equal to `Error Tolerance`. `PruningTolerance` indicates how low the edge flux ratio for a species must get before the species is pruned (removed) from the edge (regardless of edge size relative to `MaxEdgeSpeciesAfterPruning`). The value should be (significantly) lower than `Error Tolerance`. `MinSpeciesForPruning` indicates the minimum total number of species (edge and core) that must be present for pruning (due to low edge flux ratio relative to `PruningTolerance` or due to `MaxEdgeSpeciesAfterPruning`) to occur. `MaxEdgeSpeciesAfterPruning` indicates the upper limit for the size of the edge. Pruning will continue until the edge is at least this small, regardless of `PruningTolerance` (though lowest fluxes are pruned first).

When using pruning, RMG will not prune unless all reaction systems reach the goal reaction time or conversion without first exceeding the termination tolerance. Therefore, you may find that RMG is not pruning even though the model edge size exceeds `MaxEdgeSpeciesAfterPruning`. In order to increase the likelihood of pruning in such cases, you can try increasing `TerminationTolerance` to an arbitrarily high value. Alternatively, if you are using a conversion goal, because reaction systems may reach equilibrium below the goal conversion, it may be helpful to reduce the goal conversion or switch to a goal reaction time.

4.15.2 Simulator Tolerances

Fields are **Required**

The next two lines specify the absolute and relative tolerance for the ODE solver, respectively. Common values for the absolute tolerance are 1e-15 to 1e-25. Relative tolerance is usually 1e-4 to 1e-8:

```
Atol: 1e-20
Rtol: 1e-4
```

Note: You have the option of using sensitivity analysis only if you have the DASPK solver and the DAEPACK library. If you are using DASSL solver then skip the Sensitivity Analysis section.

One option when dealing with NegativeConcentrationException issues is to use the "non-negative" option with DASSL. This option is requested with the line `DynamicSimulator: DASSL: non-negative`. See the FAQ for further details.

4.16 Sensitivity Analysis

Field is **Optional**

If DASPK is chosen for the dynamic solver, RMG can perform sensitivity analysis and generate error bars on the concentration profiles. As mentioned above, however, RMG uses DASPK in a way that requires the use of the proprietary library DAEPACK which provides automatic differentiation capabilities for the sensitivity analysis (see *installation instructions* (page 28) for details). If you do not have access to this library, an alternative is to perform sensitivity analysis in CHEMKIN, using the ''chem.inp" file produced by RMG.

The sensitivity analysis is based upon first-order sensitivity coefficients and uncertainties in the rate constants. To use the sensitivity analysis, either the first or second line must be set to `on`. If both `Error bars` and `Display sensitivity coefficients` are set to off, then RMG will not perform a sensitivity analysis.

When the option to generate error bars is turned on, RMG calculates the upper and lower bounds on the concentration profiles for all core species. These upper and lower bounds are generated by the first-order sensitivity equation:

$$\ln C_i\left(t, \overrightarrow{k} + \Delta\overrightarrow{k}\right) \approx \ln C_i\left(t, \overrightarrow{k}\right) + \sum_j \frac{\partial \ln C_i}{\partial \ln k_j} \Delta \ln k_j$$

where C_i is the concentration of the i^{th} species, and $\overrightarrow{k}$ is the vector of rate constants.

If the option to display sensitivity coefficients is turned on, you must specify a list of species for which the sensitivities should be displayed. The output file will then include the sensitivities of those species with respect to all the rate constants, all the heats of formations, and the initial concentrations of all the reactants. Additionally, if you choose to display the sensitivity coefficients, RMG will print a list of the five reactions that contribute the most to the uncertainty in the concentration (for each species selected by the user).

Sensitivities to rate constants are normalized sensitivities ($\partial \ln C_i / \partial \ln k_j$), and sensitivities to heats of formation are semi-normalized ($\partial \ln C_i / \partial \Delta H_f$). The contribution of a reaction to the uncertainty is estimated as the product of the normalized sensitivity and the uncertainty in the rate constant ($\Delta \ln k$). In the following example, RMG will perform a sensitivity analysis. Error bars will be generated for all the species, and the sensitivity coefficients will be generated for carbon monoxide, carbon dioxide, methane, and water:

```
Error bars: On
Display sensitivity coefficients: On
Display sensitivity information for:
CO
CO2
CH4
H2O
END
```

Please note that the keyword `END` must be placed at the end of the Error bars section.

4.17 Primary Kinetic Library

Field is **Required** but can be empty.

The next section of the `condition.txt` file specifies which, if any, primary kinetic libraries should be used. When a reaction is specified in a primary kinetic library, RMG will use these kinetics instead of the kinetics located in the `RMG_database/kinetics_groups/<RxnFamily>/rateLibrary.txt` directory.

For details of the kinetics libraries included with RMG that can be used as a primary kinetic library, see *Kinetics Libraries* (page 77). You can specify your own primary kinetic library in the location section. In the following example, the user has created her own primary kinetic library with a few additional reactions specific to n-butane, and these reactions are to be used in addition to the Leeds' oxidation library:

```
PrimaryKineticLibrary:
Name: nbutane
Location: nbutane
Name: Leeds
Location: combustion_core/version5
END
```

If a user wished to use the GRI-Mech 3.0 mechanism as a primary kinetic library, the syntax is:

```
Name: GRI-Mech 3.0
Location: GRI-Mech3.0
```

The libraries are stored in `$RMG/databases/RMG_database/kinetics_libraries/` and the *Location:* should be specified relative to this path.

Please note that the keyword `END` must be placed at the end of the Primary Kinetic Library section. Because the units for the Arrhenius parameters are given in each library, the different libraries can have different units.

If the same reaction occurs more than once in a single Primary Kinetics Library, all instances will be recognized by RMG and will be reported as DUPLICATES in the `chem.inp` file.

If the same reaction occurs more than once in the combined library, the instance of it from the first library in which it appears is the one that gets used.

Note: RMG will not handle irreversible reactions correctly, if supplied in a Primary Kinetic Library, see *RMG "Primary Kinetic Libraries" and "Reaction Libraries" do not handle irreversible reactions properly* (page 149).

Changed in version 3.2: In previous versions this feature was called "PrimaryReactionLibrary" it has been renamed to "PrimaryKineticLibrary" to better represent its functionality.

4.18 Primary Reaction Library

Field is **Required** but can be empty.

The next section of the `condition.txt` file specifies which, if any, Reaction Libraries should be used. When a reaction library is specified, RMG will first use the reaction library to generate all the relevant reactions for the species in the core before going through the reaction templates. This feature can be used as a Primary Kinetic Library, in addition to specifying reactions that the user wants to consider while building the mechanism but that do not otherwise match any of RMG reaction family templates. Unlike the Seed Mechanism, reactions present in a Reaction Library will not be included in the core automatically from the start.

For details of the kinetics libraries included with RMG that can be used as a reaction library, see *Kinetics Libraries* (page 77). You can specify your own reaction library in the location section. In the following example, the user has created a reaction library with a few additional reactions specific to n-butane, and these reactions are to be used in addition to the Glarborg C3 library:

```
ReactionLibrary:
Name: nbutane
Location: nbutane
Name: Glarborg
Location: Glarborg/C3
END
```

The reaction libraries are stored in `$RMG/databases/RMG_database/kinetics_libraries/` and the *Location:* should be specified relative to this path.

Please note that the keyword `END` must be placed at the end of the Reaction Library section. Because the units for the Arrhenius parameters are given in each mechanism, the different mechanisms can have different units.

Note: While using a Reaction Library the user must be careful enough to provide all instances of a particular reaction in the library file, as RMG will ignore all reactions generated by its templates. For example, suppose you supply the Reaction Library with butyl_1 –> butyl_2. Although RMG would find two unique instances of this reaction (via a three- and four-member cyclic Transition State), RMG would only use the rate coefficient supplied by you in generating the mechanism.

RMG will not handle irreversible reactions correctly, if supplied in a Reaction Library, see *RMG "Primary Kinetic Libraries" and "Reaction Libraries" do not handle irreversible reactions properly* (page 149).

4.19 Seed Mechanisms

Field is: **Required** but can be empty.

The next section of the `condition.txt` file specifies which, if any, Seed Mechanisms should be used. If a seed mechanism is passed to RMG, every species and reaction present in the mechanism will be placed into the core, in addition to the species that are listed in the *Reactants* (page 37) section.

For details of the kinetics libraries included with RMG that can be used as a seed mechanism, see *Kinetics Libraries* (page 77).

You can specify your own seed mechanism in the location section. Please note that the oxidation library should not be used for pyrolysis models. The syntax for the seed mechanisms is similar to that of the primary reaction libraries, except for the `GenerateReactions` line, explained below.:

```
SeedMechanism:
Name: GRI-Mech 3.0
Location: GRI-Mech3.0
GenerateReactions: yes
Name: Leeds
Location: combustion_core/version5
GenerateReactions: yes
END
```

The seed mechanisms are stored in `$RMG/databases/RMG_database/kinetics_libraries/` and the *Location:* should be specified relative to this path.

There is a new required `GenerateReactions` line in seed mechanisms that controls how RMG adds the seed species and reactions to the model core. If set to `yes`, RMG will use its reaction families to react all seed species with one another; the generated reactions will supplement the seed reactions. If set to `no`, RMG will not generate reactions of the seed species. In either case, RMG will react the species in the condition file with one another and with all species in the seed mechanism.

Please note that the keyword `END` must be placed at the end of the Seed Mechanism section. Because the units for the Arrhenius parameters are given in each mechanism, the different mechanisms can have different units. Additionally, if the same reaction occurs more than once in the combined mechanism, the instance of it from the first mechanism in which it appears is the one that gets used.

Note: For more information on what files to include in a Primary Kinetic / Reaction Library or Seed Mechanism, visit *Building a Primary Kinetic Library / Reaction Library / Seed Mechanism* (page 103)

4.20 Chemkin Units

Field is **Required**

The last section of the `condition.txt` file specifies the units for the Arrhenius A and Ea parameters reported in the `chem.inp` file. The acceptable units for A are "moles" and "molecules"; the acceptable units for Ea are "kcal/mol", "cal/mol", "kJ/mol", "J/mol", and "Kelvins":

```
ChemkinUnits:
A: moles
Ea: kcal/mol
```

Another option available to the user is to specify whether the comments following each set of Arrhenius parameters in the `chem.inp` file are concise or verbose. The verbose comments describe how the reported A, n, and Ea Arrhenius parameters were calculated in the event RMG could not find an exact match for the functional groups.:

```
ChemkinUnits:
Verbose: on
A: moles
Ea: kcal/mol
```

If the Verbose field is not present, RMG will default to "off".

Note: The `chem.inp` file generated with the Verbose field turned "on" may have a comment that spans hundreds of characters. These verbose comments may cause the CHEMKIN interpreter to throw an error when running the Pre-Processor.

CHAPTER

FIVE

LIQUID PHASE SYSTEMS

To simulate liquids in RMG requires three components.

After the `PressureModel` line in the `condition.txt` input file, add a section like this:

```
EquationOfState: Liquid
Solvation: on octane
Diffusion: on 1.0e-3
```

5.1 Equation of state

The first statement `EquationOfState: Liquid` will have two effects:

1. disable the ideal gas law renormalization and instead rely on the concentrations you specified in the input file to initialize the system.
2. prevent the volume from changing when there is a net stoichiometry change due to a chemical reaction (A = B + C).

5.2 Solvation thermochemistry

The next correction for liquids is solvation effects on the thermochemistry. The method used is described briefly in "Building Models of Liquid Phase Kinetics and Separation: Hydrocarbon Autoxidation", Richard H. West, Amrit Jalan, and William H. Green, AIChE Annual Meeting (http://aiche.confex.com/aiche/2010/webprogram/Paper200216.html), 2010 (download pdf (http://web.mit.edu/rwest/www/Richard%20West%20AIChE%20Extended%20Abstract%20153g.pdf)) and in more detail in "An Extensible Framework for Capturing Solvent Effects in Computer Generated Kinetic Models", Amrit Jalan, Richard H. West, and William H. Green, *J. Phys. Chem. B*, 2013, 117 (10), pp 2955–2970 DOI: 10.1021/jp310824h (http://dx.doi.org/10.1021/jp310824h)

Warning: In versions of RMG prior to 4.0.1, there was an error in predicting solvation thermochemistry at temperatures other than 298 K. Please read the release notes to 4.0.1 and upgrade to a recent version of RMG.

The statement `Solvation: on <solvent>` turns this on, and specifies the solvent parameters to use. Parameters are provided for the following solvents: acetonitrile benzene butanol carbontet chloroform cyclohexane decane dibutylether dichloroethane dimethylformamide dimethylsulfoxide dodecane ethanol ethylacetate heptane hexadecane hexane isooctane nonane octane octanol pentane toluene undecane water.

5.3 Diffusion-limited kinetics

The next correction for liquid-phase reactions is to ensure that bimolecular reactions do not exceed their diffusion limits. The diffusivity is estimated for each species using a Stokes-Einstein hard sphere model. The viscosity of the solvent is specified in units of Pa.s using the command `Diffusion: on <viscosity>`

To build accurate models of liquid phase chemical reactions you will also want to modify your kinetics libraries.

5.4 Example liquid-phase condition file

This is an example of a small condition file for a liquid-phase system:

```
// autoxidation of octane


Database: RMG_database

PrimaryThermoLibrary:
Name: GRIMech3.0
Location: GRI-Mech3.0
Name: RMG-minimal
Location: primaryThermoLibrary
END

PrimaryTransportLibrary:
Name: GRIMech3.0
Location: GRI-Mech3.0
END

ReadRestart: no
WriteRestart: no

TemperatureModel: Constant (K) 500
PressureModel: Constant (atm) 1

/// THESE ARE THE IMPORTANT PARTS FOR LIQUID SIMULATIONS
EquationOfState: Liquid
Solvation: on octane
Diffusion: on 1.0e-3

InitialStatus:
```

```
Octane (mol/cm3) 6.154e-3
1 C 0 {3,S}
2 C 0 {4,S}
3 C 0 {1,S} {5,S}
4 C 0 {2,S} {6,S}
5 C 0 {3,S} {7,S}
6 C 0 {4,S} {8,S}
7 C 0 {5,S} {8,S}
8 C 0 {6,S} {7,S}

O2 (mol/cm3) 4.953e-6  ConstantConcentration
1 O 1 {2,S}
2 O 1 {1,S}

END

InertGas:
N2 (mol/cm3) 0.0
Ar (mol/cm3) 0.0
END

SpectroscopicDataEstimator: off
PressureDependence: off

// Change Goal ReactionTime:  10 (min) for a more realistic simulation
FinishController:
(1) Goal ReactionTime:  10 (sec)
(2) Error Tolerance: 0.5

DynamicSimulator: DASSL
TimeStep: AUTO
Atol: 1e-18
Rtol: 1e-8

PrimaryKineticLibrary:
END

ReactionLibrary:
END

SeedMechanism:
END

ChemkinUnits:
A: moles
Ea: kcal/mol
```

CHAPTER

SIX

EXAMPLE CONDITION FILES

6.1 Ethane pyrolysis (Minimal)

This is the minimal example file characterizing a very basic system for ethane pyrolysis and should run quickly if RMG is set up properly. It does not include any calculation of pressure-dependent reaction rates.

```
//tracks the decomposition of pure ethane, without any pressure-dependent reactions.

Database: RMG_database

PrimaryThermoLibrary:
Name: RMG-minimal
Location: primaryThermoLibrary
END

PrimaryTransportLibrary:
Name: GRIMech3.0
Location: GRI-Mech3.0
END

ReadRestart: no
WriteRestart: no

TemperatureModel: Constant (K) 1350
PressureModel: Constant (atm) 1

InitialStatus:

C2H6 (mol/cm3) 1.0
1 C 0 {2,S}
2 C 0 {1,S}

END

InertGas:
N2 (mol/cm3) 0.0
Ar (mol/cm3) 0.0
END
```

```
SpectroscopicDataEstimator: off
PressureDependence: off

FinishController:
(1) Goal Conversion: C2H6 0.9
(2) Error Tolerance: 0.1

DynamicSimulator: DASSL
Conversions: AUTO
Atol: 1e-18
Rtol: 1e-8

PrimaryKineticLibrary:
END

ReactionLibrary:
END

SeedMechanism:
END

ChemkinUnits:
A: moles
Ea: kcal/mol
```

6.2 1,3-hexadiene pyrolysis

This example models the pyrolysis of 1,3-hexadiene and demonstrates the effect of turning on the pressure-dependence module within RMG.

```
//tracks the consumption of 1,3-hexadiene in presence of N2, Methane and hydrogen.
//notice the primary reaction library is turned off because this is not
//a oxidation mechanism. Also the sensitivity analysis section is missing
//because we are using the dassl solver.

//This example should take roughly 2-3 minutes to run to completion.


Database: RMG_database

//MaxCarbonNumberPerSpecies:
//MaxOxygenNumberPerSpecies:
//MaxRadicalNumberPerSpecies:
//MaxSulfurNumberPerSpecies:
//MaxSiliconNumberPerSpecies:
//MaxHeavyAtomPerSpecies:

PrimaryThermoLibrary:
Name: GRIMech3.0
Location: GRI-Mech3.0
Name: RMG-minimal
```

```
Location: primaryThermoLibrary
END

PrimaryTransportLibrary:
Name: GRIMech3.0
Location: GRI-Mech3.0
END

ReadRestart: no
WriteRestart: yes

TemperatureModel: Constant (K) 1350
PressureModel: Constant (atm) 1

InitialStatus:

HXD13 (mol/cm3) 6.829e-4
1 C 0 {2,D}
2 C 0 {1,D} {3,S}
3 C 0 {2,S} {4,D}
4 C 0 {3,D} {5,S}
5 C 0 {4,S} {6,S}
6 C 0 {5,S}

CH4 (mol/cm3) 0.104
1 C 0

H2 (mol/cm3) 1.56e-2
1 H 0 {2,S}
2 H 0 {1,S}

END

InertGas:
N2 (mol/cm3) 0.8797
Ar (mol/cm3) 0.0e-6
END

SpectroscopicDataEstimator: FrequencyGroups
PressureDependence: ModifiedStrongCollision
PDepKineticsModel: Chebyshev 6 4
TRange: (K) 300.0 2000.0 8
PRange: (bar) 0.01 100.0 5
DecreaseGrainSize: yes

FinishController:
(1) Goal Conversion: HXD13 0.9
(2) Error Tolerance: 0.5

DynamicSimulator: DASSL
Conversions: AUTO
Atol: 1e-18
Rtol: 1e-8
```

```
PrimaryKineticLibrary:
//Name: RMG-example
//Location: Example
END

ReactionLibrary:
//Name: GRIMech3.0
//Location: GRI-Mech3.0
END


SeedMechanism:
//Name: Leeds
//Location: combustion_core/version5
//GenerateReactions: yes
//Name: GRIMech3.0
//Location: GRI-Mech3.0
//GenerateReactions: yes
END

ChemkinUnits:
A: moles
Ea: kcal/mol
```

6.3 Butane oxidation with pruning

This example illustrates the use of pruning to reduce model generation time, as well as the simulation of as multiple reaction conditions in a single condition file. The example should take at least several hours to run and may require allocation of a large amount of memory (e.g. 1500 MB) to complete:

```
//tracks the consumption of Butane in presence of O2.

//This example illustrates the use of pruning, as well
//as multiple reaction conditions. The example should
//take at least several hours to run and may require allocation
//of a large amount of memory (e.g. 1500 MB) to complete.

Database: RMG_database

//MaxCarbonNumberPerSpecies:
//MaxOxygenNumberPerSpecies:
//MaxRadicalNumberPerSpecies:
//MaxSulfurNumberPerSpecies:
//MaxSiliconNumberPerSpecies:
//MaxHeavyAtomPerSpecies:

PrimaryThermoLibrary:
Name: GRIMech3.0
Location: GRI-Mech3.0
Name: RMG-minimal
```

```
Location: primaryThermoLibrary
END

PrimaryTransportLibrary:
Name: GRIMech3.0
Location: GRI-Mech3.0
END

ForbiddenStructures:


END

ReadRestart: no
WriteRestart: yes

TemperatureModel: Constant (K) 800 1000 2000
PressureModel: Constant (atm)  20 30 40

InitialStatus:

C4H10 (mol/cm3)   1     2
1 C 0 {2,S}
2 C 0 {1,S} {3,S}
3 C 0 {2,S} {4,S}
4 C 0 {3,S}

O2 (mol/cm3)     6.5   5.5
1 O 1 {2,S}
2 O 1 {1,S}

END

InertGas:
N2  (mol/cm3)     24.399  24.399
Ar  (mol/cm3)        0      0
END

SpectroscopicDataEstimator: off
PressureDependence: off
//PressureDependence: ModifiedStrongCollision
//PDepKineticsModel: Chebyshev

FinishController:
(1) Goal Conversion: C4H10 0.7
(2) Error Tolerance: 0.5

DynamicSimulator: DASSL
//Conversions: AUTO
Conversions: AUTOPRUNE
TerminationTolerance: 1000
PruningTolerance: 1.0E-18
MinSpeciesForPruning: 1000
```

```
MaxEdgeSpeciesAfterPruning: 10000
Atol: 1e-18
Rtol: 1e-8

PrimaryKineticLibrary:
//Name: RMG-example
//Location: RMG_database/primaryReactionLibrary/Example
END

ReactionLibrary:

END

SeedMechanism:
//Name: Leeds
//Location: RMG_database/SeedMechanisms/combustion_core/version5
//GenerateReactions: yes
Name: GRIMech3.0
Location: GRI-Mech3.0
GenerateReactions: yes
//Name: Glarborg
//Location: RMG_database\SeedMechanisms\Glarborg\C3_light
//GenerateReactions: yes

END

ChemkinUnits:
A: moles
Ea: kcal/mol
```

6.4 Cyclopropane pyrolysis with quantum mechanics calculations

This is an example illustrating the use of on-the-fly thermo calculations. Gaussian03 is used to estimate thermodynamic properties of cyclic species, like cyclopropane. In particular, the semi-empirical PM3 method, with RRHO treatment of partition functions is used. Without this feature, RMG would try to estimate thermodynamic properties of cyclic species using the typical Benson groups, and would only apply an appropriate ad hoc ring correction if it is in Ring_Library.txt. The example should take roughly 45 minutes to run and requires several additional dependencies, as described in the *installation documentation* (page 29).

```
//This is an example illustrating the use of on-the-fly thermo
//calculations. Gaussian03 is used to estimate thermodynamic
//properties of cyclic species, like cyclopropane. In particular,
//the semi-empirical PM3 method, with RRHO treatment of partition
//functions is used. Without this feature, RMG would try to estimate
//thermodynamic properties of cyclic species using the typical Benson
//groups, and would only apply an appropriate ad hoc ring correction
//if it is in Ring_Library.txt. The example should take roughly 45
//minutes to run and requires several additional dependencies, as
//described in the documentation.

Database: RMG_database
```

```
//MaxCarbonNumberPerSpecies:
//MaxOxygenNumberPerSpecies:
//MaxRadicalNumberPerSpecies:
//MaxSulfurNumberPerSpecies:
//MaxSiliconNumberPerSpecies:
//MaxHeavyAtomPerSpecies:

PrimaryThermoLibrary:
Name: GRIMech3.0
Location: GRI-Mech3.0
Name: RMG-minimal
Location: primaryThermoLibrary
END

PrimaryTransportLibrary:
Name: GRIMech3.0
Location: GRI-Mech3.0
END

ReadRestart: no
WriteRestart: no

TemperatureModel: Constant (K) 1350
PressureModel: Constant (atm) 1

//thermo strategy? BensonOnly/QMforCyclics/Hybrid
ThermoMethod: QMforCyclics Gaussian03
MaxRadNumForQM: 0
CheckConnectivity: check
KeepQMFiles: no

InitialStatus:

Cyclopropane (mol/cm3) 6.829e-4
1 C 0 {2,S} {3,S}
2 C 0 {1,S} {3,S}
3 C 0 {1,S} {2,S}

END

InertGas:
N2 (mol/cm3) 0.8797
Ar (mol/cm3) 0.0e-6
END

SpectroscopicDataEstimator: off
PressureDependence: off

FinishController:
(1) Goal Conversion: Cyclopropane 0.9
(2) Error Tolerance: 0.1

DynamicSimulator: DASSL
```

```
Conversions: AUTO
Atol: 1e-18
Rtol: 1e-8

PrimaryKineticLibrary:
END

ReactionLibrary:
END

SeedMechanism:
END

ChemkinUnits:
A: moles
Ea: kcal/mol
```

6.5 Octane liquid phase autoxidation

This example showcases RMG's ability to model liquid-phase reaction systems.

```
// autoxidation of octane


Database: RMG_database

PrimaryThermoLibrary:
Name: GRIMech3.0
Location: GRI-Mech3.0
Name: RMG-minimal
Location: primaryThermoLibrary
END

PrimaryTransportLibrary:
Name: GRIMech3.0
Location: GRI-Mech3.0
END

ReadRestart: no
WriteRestart: no

TemperatureModel: Constant (K) 500
PressureModel: Constant (atm) 1

/// THESE ARE THE IMPORTANT PARTS FOR LIQUID SIMULATIONS
EquationOfState: Liquid
Solvation: on octane
Diffusion: on 1.0e-3

InitialStatus:
```

```
Octane (mol/cm3) 6.154e-3
1 C 0 {3,S}
2 C 0 {4,S}
3 C 0 {1,S} {5,S}
4 C 0 {2,S} {6,S}
5 C 0 {3,S} {7,S}
6 C 0 {4,S} {8,S}
7 C 0 {5,S} {8,S}
8 C 0 {6,S} {7,S}

O2 (mol/cm3) 4.953e-6  ConstantConcentration
1 O 1 {2,S}
2 O 1 {1,S}

END

InertGas:
N2 (mol/cm3) 0.0
Ar (mol/cm3) 0.0
END

SpectroscopicDataEstimator: off
PressureDependence: off

// Change Goal ReactionTime:  10 (min) for a more realistic simulation
FinishController:
(1) Goal ReactionTime:  10 (sec)
(2) Error Tolerance: 0.5

DynamicSimulator: DASSL
TimeStep: AUTO
Atol: 1e-18
Rtol: 1e-8

PrimaryKineticLibrary:
END

ReactionLibrary:
END

SeedMechanism:
END

ChemkinUnits:
A: moles
Ea: kcal/mol
```

6.6 Diethylsulfide and H2O desulfurization

RMG 4.0+ can now handle systems involving sulfur-containing compounds. This example models the desulfurization of diethylsulfide in the presence of water.

```
// Diethylsulfide + H2O desulfurization

Database: RMG_database

MaxCarbonNumberPerSpecies: 18
MaxOxygenNumberPerSpecies: 3
MaxSulfurNumberPerSpecies: 3

PrimaryThermoLibrary:
Name: RMG_Default
Location: primaryThermoLibrary
Name: Sulfur_Thermo
Location: SulfurLibrary
Name: DFT_QCI
Location: DFT_QCI_thermo
END

PrimaryTransportLibrary:
END

ForbiddenStructures:
END

ReadRestart: No
WriteRestart: Yes

TemperatureModel: Constant (K) 673
PressureModel: Constant (Bar) 244.3

InChIGeneration: off

InitialStatus:

(1) DES (mol/cm3) 0.000363
1 C 0 {2,S}
2 C 0 {1,S} {3,S}
3 S 0 {2,S} {4,S}
4 C 0 {3,S} {5,S}
5 C 0 {4,S}

(2) H2O (mol/cm3) 0.0040
1 O 0

(3) ethanethial (mol/cm3) 0.0
1 C 0 {2,S}
2 C 0 {1,S} {3,D}
3 S 0 {2,D}

(4) hydro-ET (mol/cm3) 0.0
1 C 0 {2,S}
2 C 0 {1,S} {3,S} {4,S}
3 S 0 {2,S}
4 O 0 {2,S}
```

```
(5) hydro-ET-rad (mol/cm3) 0.0
1 C 0 {2,S}
2 C 0 {1,S} {3,S} {4,S}
3 S 1 {2,S}
4 O 0 {2,S}

(6) ethanal (mol/cm3) 0.0
1 C 0 {2,S}
2 C 0 {1,S} {3,D}
3 O 0 {2,D}

END

InertGas:
Ar (mol/cm3)        0
N2 (mol/cm3)        0
He (mol/cm3)        0
END

SpectroscopicDataEstimator: off
PressureDependence: off

FinishController:
(1) Goal Conversion: DES 0.10
(2) Error Tolerance: 0.50

DynamicSimulator: DASSL
Conversions: AUTO
Atol: 1E-18
Rtol: 1E-8

PrimaryKineticLibrary:
END


ReactionLibrary:
END

SeedMechanism:
Name: Hydrolysis
Location: Sulfur/Thial_Hydrolysis
GenerateReactions: no
END

ChemkinUnits:
Verbose: off
A: moles
Ea: kcal/mol
```

CHAPTER

SEVEN

CREATING A CONDITION FILE USING THE GRAPHICAL USER INTERFACE

A user may construct the `condition.txt` file by hand or by using the Graphical User Interface.

7.1 Launching the Graphical User Interface

First, change to the directory where you want the output files to be stored; for the following example, we will store the output in the directory `run`, located in the $RMG directory.

For Linux users, run the following commands to launch the GUI:

```
cd $RMG/run
java -Xmx500m -classpath $RMG/bin/RMG.jar GUI > output &
```

For Windows users, run the following commands to launch the GUI:

```
cd "%rmg%\run"
java -Xmx500m -classpath "%rmg%\bin\RMG.jar" GUI > output &
```

7.2 The "Initial Conditions" tab

This tab of the GUI allows the user to enter the *Reactants* (page 37), *Inert Gases* (page 38), and *Initial Conditions* (page 35) for the simulation.

7.2.1 Reactants and Inert Gases

For each species, a "Name," "Concentration," "Units," and "Reactivity" must be specified. The three options for "Reactivity" are:

- "Inert" - a species for which RMG will not build a reaction mechanism
- "Reactive-User" - a species for which RMG will build a mechanism, but only according to the list of reactions the user has provided

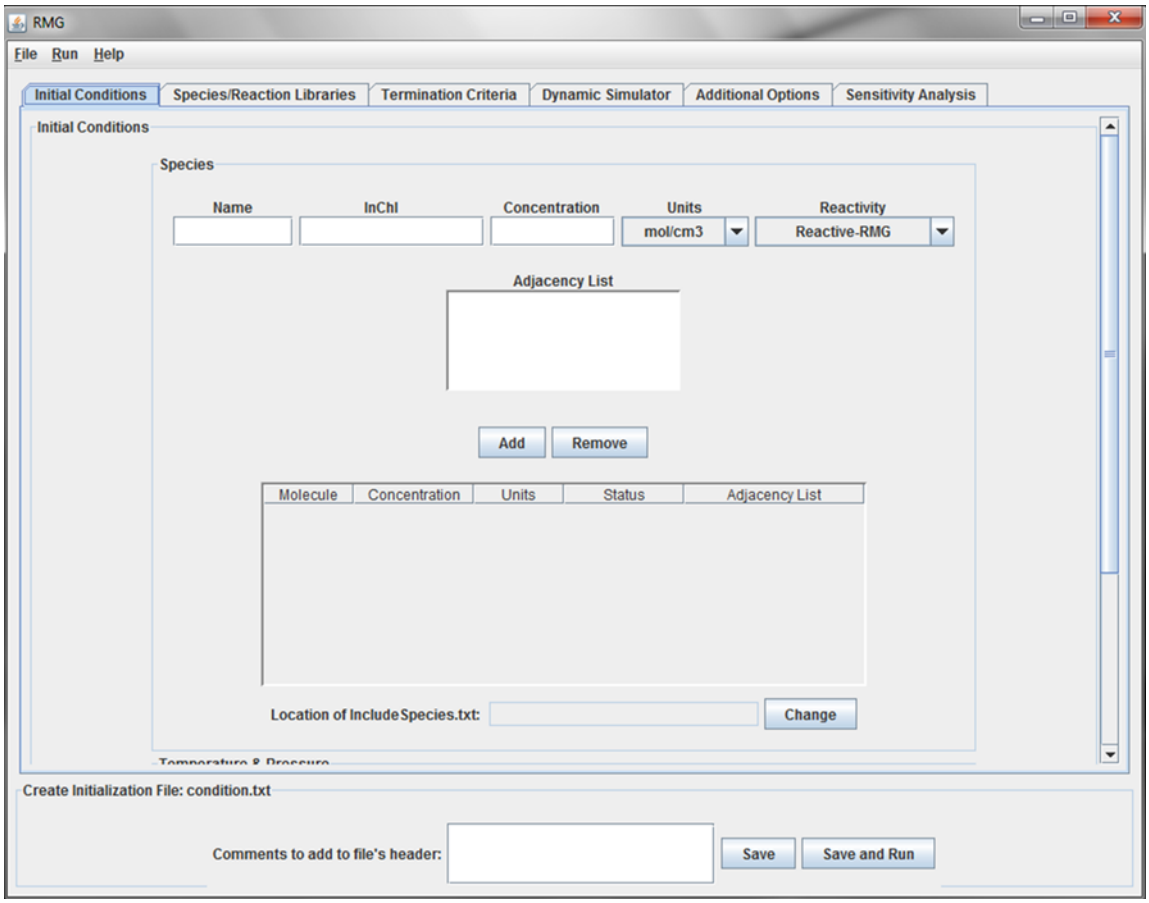


Figure 7.1: Initial Conditions tab

- "Reactive-RMG" - a species for which RMG will build a mechanism, according to the reaction family templates located in the `RMG_database/kinetics` folder

If the species is selected as "Inert", no structural information for the molecule is necessary. RMG only recognizes the following Inerts: N2, He, Ne, Ar.

If the species is selected as either "Reactive-User" or "Reactive-RMG", the user must supply structural information about the molecule. This information may be provided in one of three ways: adjacency list, InChI, or .mol file.

An example of an adjacency list (for methane) is:

```
1 C 0 {2,S} {3,S} {4,S} {5,S}
2 H 0 {1,S}
3 H 0 {1,S}
4 H 0 {1,S}
5 H 0 {1,S}
```

An equally valid adjacency list for methane (with the Hydrogen atoms removed) is:

```
1 C 0
```

If the user knows the InChI of the molecule (and the InChI executable is located in the `RMG/bin/` folder), the GUI will convert the InChI to an adjacency list upon pushing the "Add" button. If using either InChI software version 1.01 or software version 1.02beta, the InChI for methane is:

```
InChI=1/CH4/h1H4
```

However, if using the InChI software version 1.02 for Standard InChI/InChIKey, the InChI for methane is:

```
InChI=1S/CH4/h1H4
```

Please make sure the InChI you pass to the RMG GUI is consistent with the InChI version located in the `$RMG\bin\` folder. Upon pushing the "Add" button, if neither an adjacency list nor InChI have been provided, the GUI will ask the user for the location of the .mol file associated with this species. If the InChI executable is in the proper folder, the GUI will convert the .mol file to an adjacency list.

The user may add/remove species to the model by clicking the appropriate buttons.

7.2.2 Reactive-User Species

If a "Reactive-User" species is added to the model, the GUI will ask the user for the location of the `IncludeSpecies.txt` file. If no "Reactive-User" species are added to the model, the "Location of IncludeSpecies.txt" field will remain inactive.

7.2.3 Initial Conditions

RMG currently only has one pressure/temperature model: "Constant". However, the user may specify a list of temperatures and pressures for which to build the reaction mechanism. The list of temperatures (or pressures) should be separated from one another with a blank space or a return carriage.

7.3 The "Species/Reactions Libraries" tab

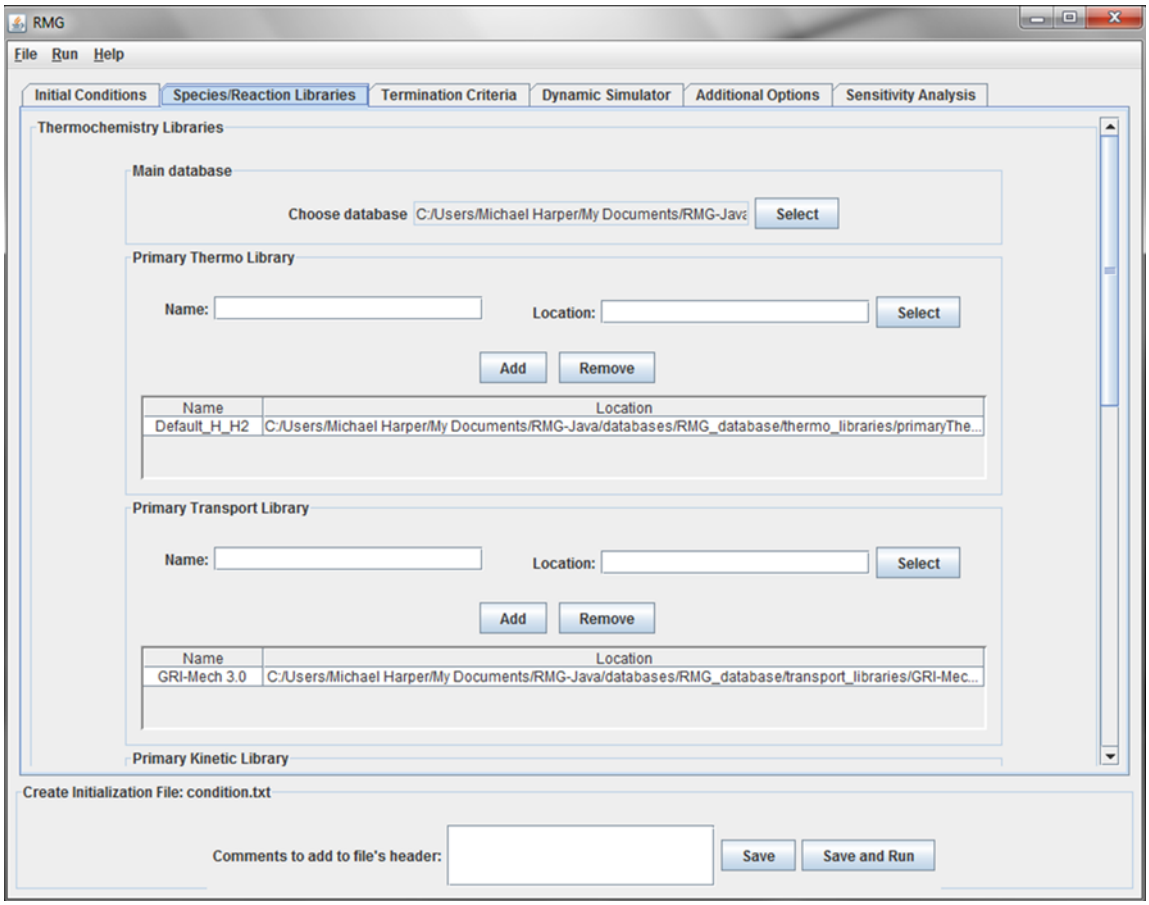


Figure 7.2: Species/Reactions Libraries tab

This tab of the GUI allows the user to enter the location of the thermochemical libraries RMG will use in building the model.

The default location of the *Main Database* (page 32) is the `RMG/database/RMG_database` folder. We recommend to use the default settings.

The default location of the *Primary Thermo Library* (page 33) is the `RMG/database/RMG_database/thermo_libraries/primaryThermoLibrary` folder. We recommend to include at least this default primary thermochemistry library. To add additional primary thermo libraries: Supply a Name, select the directory by pressing the "Select" button, and then push "Add".

The default *Primary Transport Library* (page 33) is the transport properties reported in the GRI-Mech 3.0 mechanism. If running a pressure-dependent simulation, we recommend including a library that contains small molecules, i.e. C0-C2 species, as the RMG group-additivity approach is more accurate for larger molecules.

To add a *Primary Kinetics Library* (page 45), a *Primary Reaction Library* (page 46), or a *Seed Mechanism* (page 47): Supply a Name, select the directory by pressing the "Select" button, and then push "Add".

Note: The order of the "Primary Thermo Libraries", "Primary Transport Library", "Primary Kinetics Libraries", "Primary Reaction Library", and "Seed Mechanisms" in the tables is important. The order established in the table will be conserved when writing the `condition.txt` file. For any reactions that appear in multiple "Primary Kinetics Libraries" (or "Seed Mechanisms"), RMG will use the information contained in the first "Primary Kinetics Library" (or "Seed Mechanism") and ignore all others. Similarly, for any species that appear in multiple "Primary Thermo Libraries", RMG will use the information contained in the first "Primary Thermo Library" and ignore all others.

7.4 The "Termination" tab

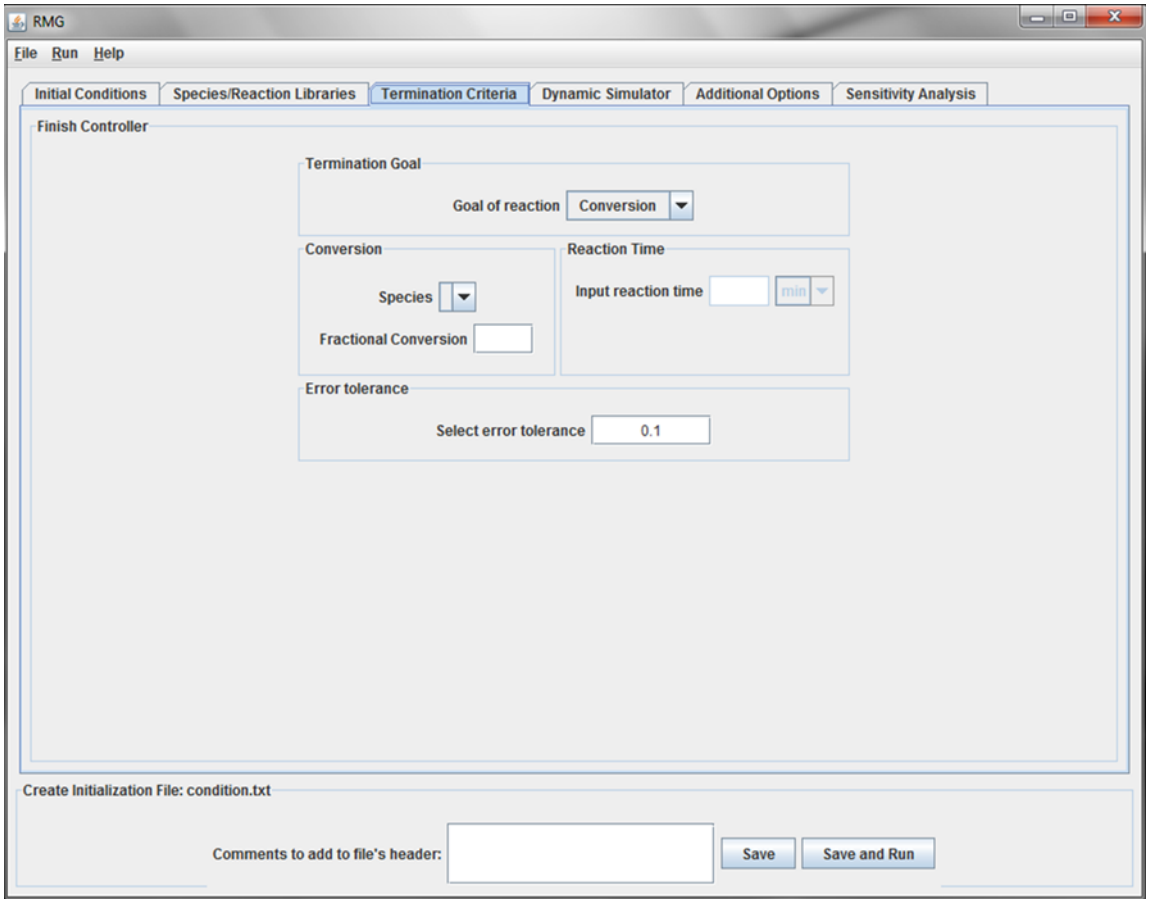


Figure 7.3: Termination tab

This tab of the GUI allows the user to specify the *termination criteria* (page 41) for the model. A user may specify either the desired "ReactionTime" or "Conversion".

If "Conversion" is selected, the user must specify a species and a fraction conversion (ranging from 0 to 1). The "Species" drop-down menu will only contain the "Reactive-RMG" and "Reactive-User" species specified in the Initialization tab.

If "ReactionTime" is selected, the user must specify the value and units of the desired reaction time.

Lastly, the user must specify an "Error tolerance" for the model. If building a mechanism for a new system, we recommend starting with a high "Error tolerance" (e.g 0.1).

7.5 The "Dynamic Simulator" tab

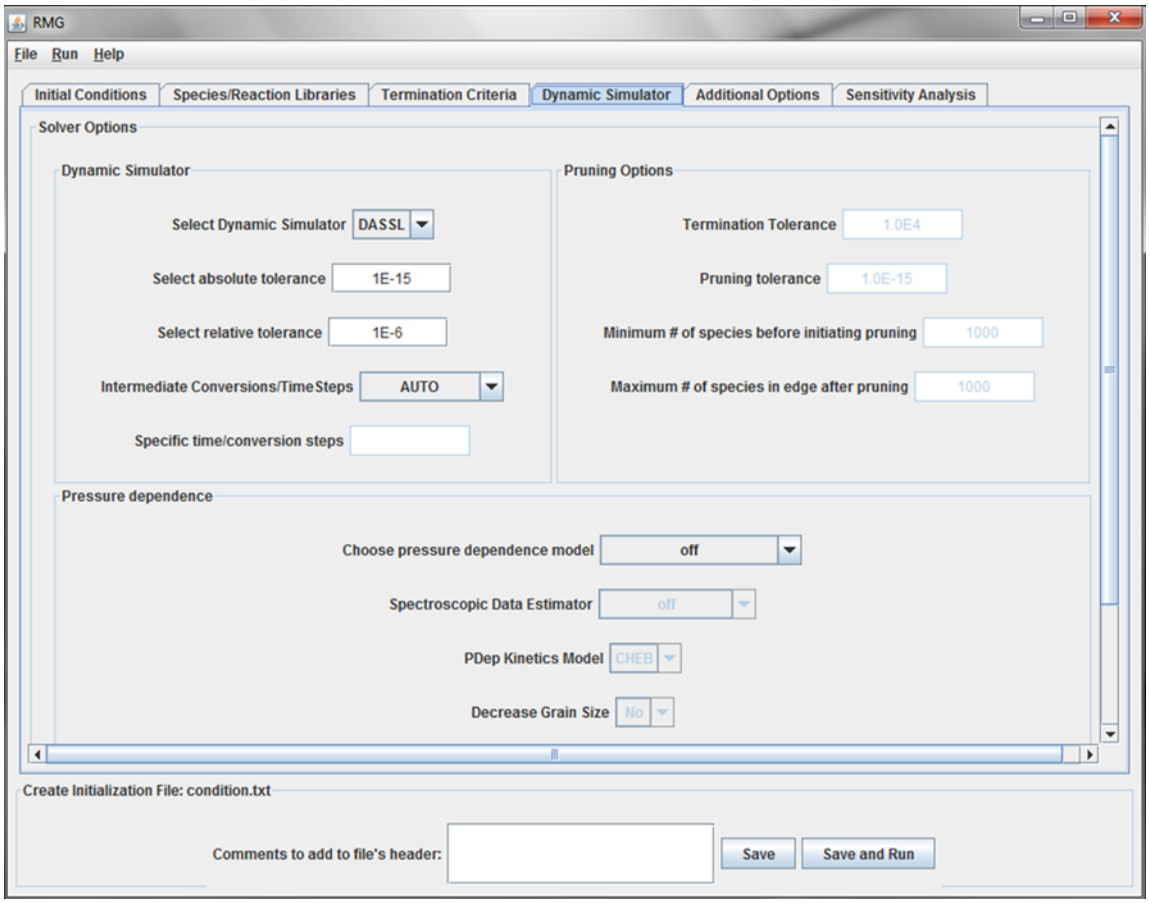


Figure 7.4: Dynamic Simulator tab

This tab of the GUI allows the user to specify options for:

- *the ODE solver* (page 42)
- *pruning* (page 43)
- *building a pressure dependent mechanism* (page 39)

7.5.1 The ODE Solver

The user must select a "Dynamic Simulator" and set the Absolute and Relative Tolerances for that Dynamic Simulator. The user may also select Intermediate times or conversions for RMG to check whether the reaction mechanism needs to be enlarged. We recommend using the automatic time stepping, which is invoked by typing "AUTO" in the "Intermediate" time/conversion fields.

Note: If the user specifies "Intermediate Time Steps," the values for the "Specific time steps" field must have units of Seconds.

7.5.2 Pruning

The user may enable the pruning options by selecting "AUTOPRUNE" in the "Intermediate Conversions/TimeSteps" field. More information for each field in the "Pruning Options" portion of the GUI is located *here* (page 43)

7.5.3 The pressure dependence model

The user must specify a *pressure dependence model* (page 39). The options are:

- off
- Modified Strong Collision
- Reservoir State

If the user specifies a pressure dependent model, the "Spectroscopic Data Estimator" and "PDep Kinetics Model" fields will be enabled. The only option for the "Spectroscopic Data Estimator" is:

- Frequency Groups

The options for the "PDep Kinetics Model" are:

- CHEB - resulting in Chebyshev polynomials as auxiliary information for pressure-dependent reactions
- PLOG - resulting in Modified Arrhenius parameters at different pressures as auxiliary information for pressure-dependent reactions
- Rate - resulting in no auxiliary information and modified Arrhenius parameters of A 0.0 0.0, where A = k(T,P)

If CHEB or PLOG is selected as the "PDep Kinetics Model", the fields near the bottom of this tab will become enabled. These fields are optional. However, if data is entered into any of the eight fields, all eight fields must be filled.

The user may also select whether additional energy grains should be included in solving the Master Equation (for pressure-dependent reactions) in the case the calculated k(T,P) is greater than the high-pressure limit rate coefficient.

7.6 The "Additional Options" tab

This tab allows the user to specify additional (and usually optional) information to RMG.

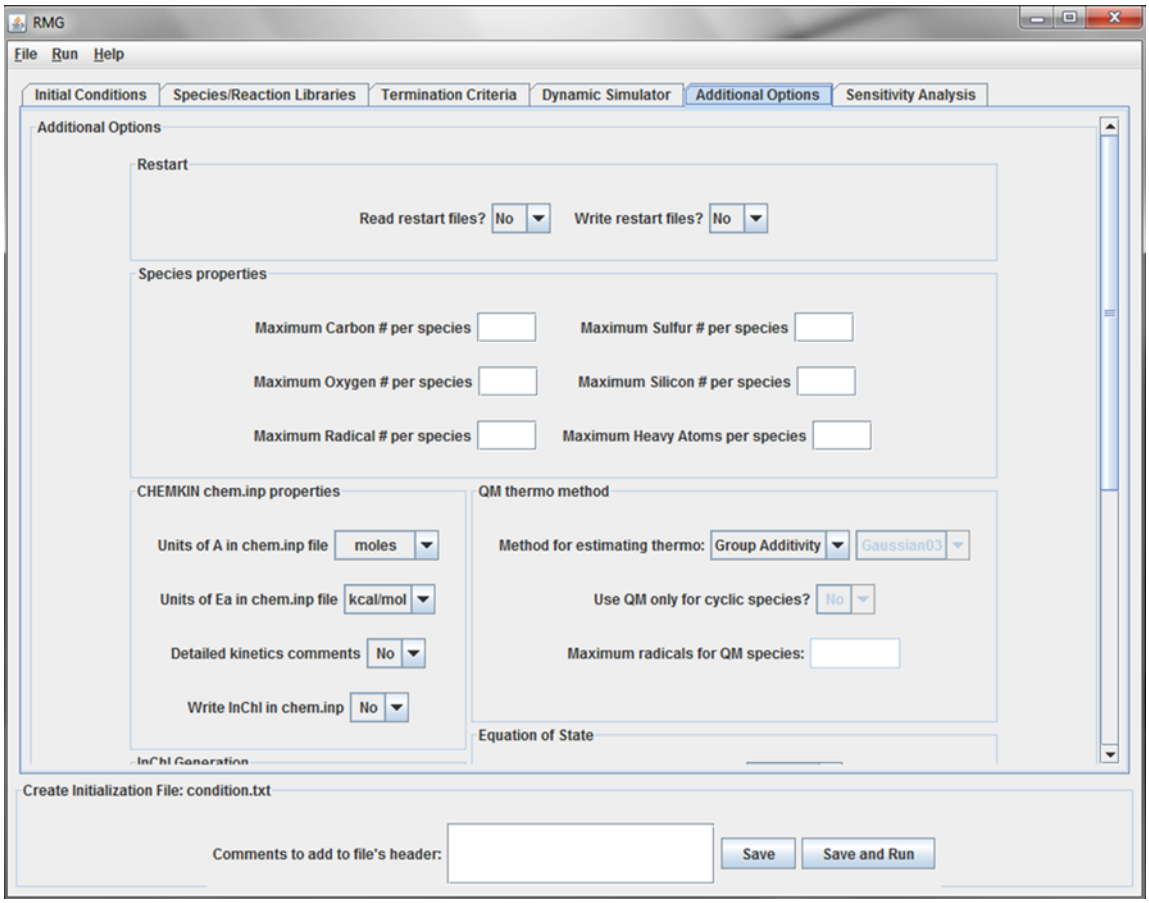


Figure 7.5: Additional Options tab

7.6.1 Restart files

The user may choose to read and/or write *Restart* (page 34) files. If the "Read restart files" option is turned on, a `Restart` folder must be present in the working directory.

7.6.2 Species properties

The user may choose to override any of the following species properties, by entering data into the appropriate field:

- Maximum carbon number per species
- Maximum oxygen number per species
- Maximum radical number per species
- Maximum sulfur number per species
- Maximum silicon number per species
- Maximum heavy atom (defined as non-hydrogen atoms) per species

The user has the option to enter data for any combination of these fields. If left blank, RMG will use the default values.

7.6.3 CHEMKIN chem.inp properties

The next set of fields allow the user to specify the units of the Arrhenius parameters 'A' and 'Ea' for the `chem.inp` file generated by RMG. The user may also specify if they want the concise or verbose comments for each reaction in the `chem.inp` file. Refer to *Chemkin options* (page 48) for more details.

7.6.4 Quantum mechanics

If the user wishes to compute species thermochemistry using something other than group additivity, e.g. *quantum mechanics* (page 36), select "QM" from the drop-down menu and the remaining fields will become active.

7.6.5 InChI generation

The user may also choose to turn off/on the generation of *InChIs* (page 36). If the user chooses to turn InChI generation on, the InChI executable (v 1.01) must be located in the `RMG/bin` folder.

7.6.6 Equation of State

The user currently has only one option for the "Equation of State" field:

- Ideal Gas

7.6.7 Forbidden Structures

If the user wishes to specify any forbidden structures, e.g. any alkene (see below), a user-defined name and the associated adjacency list must be supplied.:

```
Any_Alkene
1 C 0 {2,D}
2 C 0 {1,D}
```

7.7 The "Sensitivity Analysis" tab

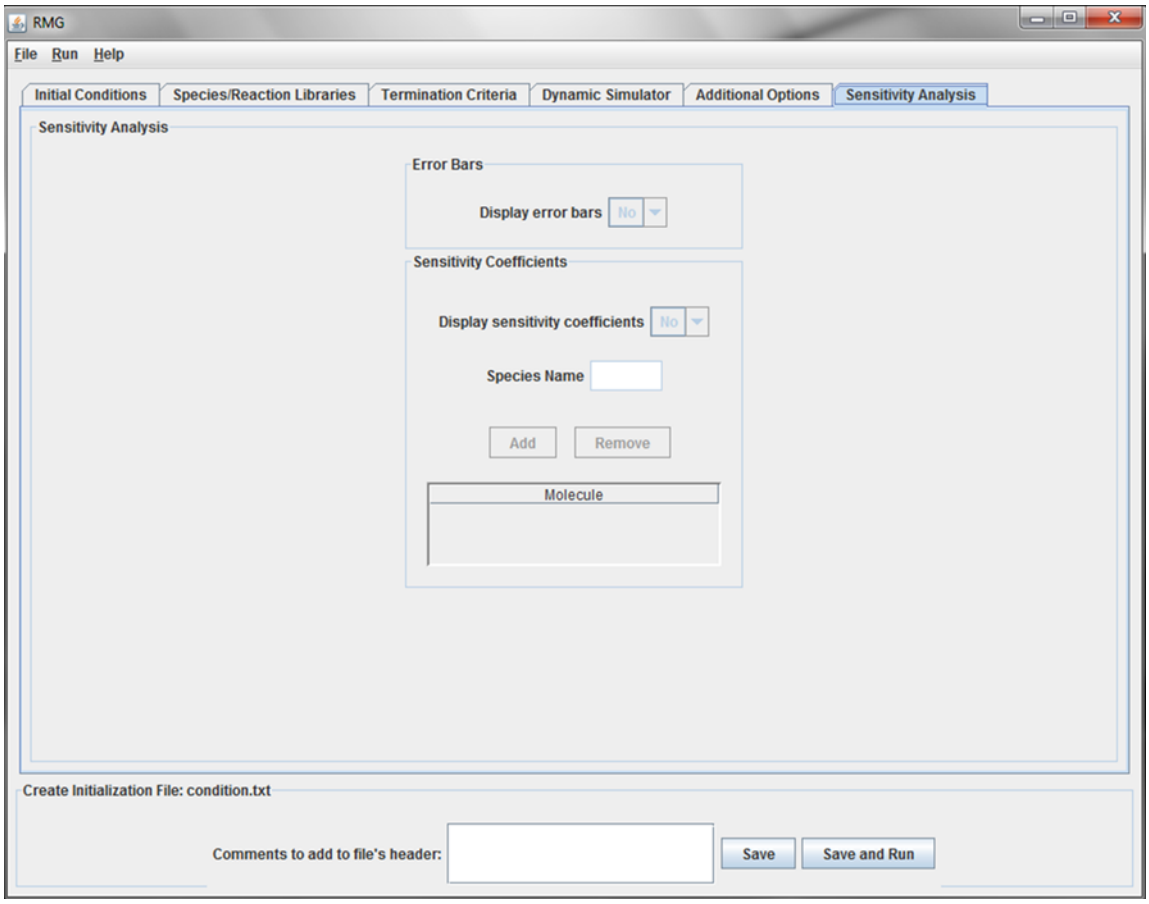


Figure 7.6: Sensitivity Analysis tab

This tab of the GUI allows the user to specify options for error bars and sensitivity coefficients. The features of this tab will only be active if "DASPK" is selected as the "Dynamic Simulator" on the *Dynamic Simulator* (page 70) tab.

The user may turn the generation of error bars "on" or "off." If turned on, upper and lower error bars will be generated for all species in the model.

The user may also turn the generation of sensitivity coefficients "on" or "off." The sensitivity coefficients will only be generated for the species listed in the "Molecule" table. The order in which the species appear

in the table is not important.

7.8 Saving the file

To save the file, the user can either press the "Save" button near the bottom of the GUI or by selecting "File" –> "Save" from the navigation toolbar. The file does not have to be named condition.txt. Also, the user may add additional comments to the `condition.txt` file by adding information to the text field at the bottom of the GUI. These comments will be added to the header of the saved file.

To save and immediately run a file, the user can press the "Save and Run" button near the bottom of the GUI.

7.9 Opening a file

To open and import the information from an already written `condition.txt` file, the user can select "File" –> "Open" from the navigation toolbar.

Note: The comments in the header of the file will not be imported.

7.10 Running the file

To run a file without viewing it, the user can select "Run" –> "Run RMG" from the navigation toolbar.

CHAPTER

EIGHT

KINETICS LIBRARIES

The seed mechanisms are stored in `$RMG/databases/RMG_database/kinetics_libraries/`

Currently, the reaction libraries that are included in RMG are:

```
the Leeds methane oxidation library
the GRI-Mech 3.0 mechanism (w/o the nitrogen-containing species and the lumped C3H7 species
the Glarborg combustion mechanism for C1 - C3 species.
```

Note: One or more species in these mechanisms would normally be forbidden by RMG, which may lead you to get an error message like "The molecular structure ... is forbidden by ...". To avoid this, use a *Primary Thermodynamic Library* (page 33) containing the species, as this will make it allowed. (e.g. the DFT_QCI_thermo library).

8.1 Leeds methane oxidation library

The species and reactions contained in the `combustion_core\version5` directory correspond to the Leeds methane oxidation mechanism, version 1.5: http://garfield.chem.elte.hu/Combustion/mechanisms/metan15.dat

8.2 GRI-Mech 3.0

The species and reactions contained in the GRI-Mech3.0 directory are those reported in the GRI-Mech 3.0 mechanism, with the exception of the following species (and any reaction they participate in):

```
Any nitrogen-containing species
C3H7
```

RMG cannot currently handle nitrogen atoms, and the C3H7 species in the GRI-Mech 3.0 mechanism is a lumped species, meant to represent n- and iso-propyl radical; RMG treats these two isomers as separate species.

8.3 Glarborg

RMG provides the option to use the rate coefficients in the master chemical mechanism developed by Peter Glarborg and coworkers at the Technical University of Denmark. The data are in the directory databases/kinetic_libraries/Glarborg. Within this directory are four subdirectories – C0, C1, C2, and C3 – where the index refers to the carbon number. The mechanisms are hierarchical in nature, so the contents of C0 are contained in C1, C1 in C2, and so forth. The C0 and C1 mechanisms include many small molecule reactions that would not necessarily fit the reaction families of RMG; consequently, they are useful Seed Mechanisms. The C2 and C3 mechanisms are optimized for high-temperature oxidation, and many of the species in the C2 and C3 mechanisms are relevant only for high-temperature chemistry. Therefore, if you are developing a mechanism for low- or moderate-temperature oxidation, it is suggested that you use C2 or C3 as a Primary Kinetic Library, rather than a Seed Mechanism.

Note: One set of kinetics in the Glarborg kinetic_library has been changed: **HO2 + OH = H2O + O2**. The reason for the change is Glarborg uses the summation of three sets of modified Arrhenius parameters to reproduce their recommended rate coefficient. However, one of the temperature-independent pre-exponential ("A") factors is negative, which neither RMG nor CHEMKIN can parse properly. Instead, the kinetics of Baulch et al. have been included.

"Evaluated kinetic data for combustion modelling. Supplement I" D.L. Baulch, C.J. Cobos, R.A. Cox, P. Frank, G. Hayman, Th. Just, J.A. Kerr, T. Murrells, M.J. Pilling, J. Troe, R.W. Walker, J. Warnatz J. Phys. Chem. Ref. Data 23 (1994) 847-1033

Details of the mechanism can be obtained in these three papers and references therein.

"Experimental measurements and kinetic modeling of CO/H-2/O-2/NO, conversion at high pressure." Rasmussen CL, Hanson J, Marshall P, Glarborg P IJCK, 2008, 40, 8, 454-480 DOI: 10.1002/kin.20327

"Experimental Measurements and Kinetic Modeling of CH4/O-2 and CH4/C2H6/O-2 Conversion at High Pressure." Rasmussen CL, Jakobsen JG, Glarborg P IJCK, 2008, 40, 12, 778-807 DOI: 10.1002/kin.20352

"Experimental and kinetic modeling study of C2H4 oxidation at high pressure." Lopez JG, Rasmussen CL, Alzueta MU, Gao Y, Marshall P, Glarborg P Proc. Comb. Inst. 2009, 32, 367-375 DOI: 10.1016/j.proci.2008.06.188

CHAPTER

NINE

RUNNING RMG

9.1 Running RMG from the Command Line in Linux

This section assumes that RMG is already installed according to the directions in Section *Linux installation* (page 24). Additionally, it is assumed that you have created a directory (either in your home directory or elsewhere) in which the initialization file `condition.txt` has been placed. In the example, we assume that we have created a directory `/hexane_pyro`, which contains the condition file `hexane_pyro.txt`.

1. Change the current working directory to the directory with the condition file:

   ```
   $ cd /hexane_pyro/
   ```

2. Run the following Java command to launch the program (all one line):

   ```
   $ java -Xmx500m -classpath $RMG/bin/RMG.jar RMG hexane_pyro.txt
   ```

When RMG is started, it creates several subdirectories: `2Dmolfiles`, `3Dmolfiles`, `chemkin`, `fame`, `frankie`, `GATPFit`, `InChI`, `ODESolver`, `Pruning`, `QMfiles` and `Restart`. By default these are in the directory from which you ran the command, but to configure them see *Configuring paths using Environment Variables* (page 80).

In this example, all of the output from RMG is piped to a file `output.log`. The `> output` part of the command is optional but recommended. The Java option `-Xmx500m` is hardware specific. In this example, we assume that we can allot five hundred megabytes of memory for Java Virtual Machine running RMG; please consult a guide to Java for more information. On unix systems we recommend allocating less than half of your available virtual memory, because of the way the Java Virtual Machine forks processes.

To see the content of the output file as it is being generated, use the command: `$ tail -f output.log`.

9.2 Running RMG from the Command Line in Windows

To run RMG in Windows: Open a DOS-prompt, change the current working directory to the directory with the condition file, and run the following Java command to launch the program. In this example, the condition file is located in a (new) folder named conditionFiles:

```
cd "%rmg%"\conditionFiles
java -Xmx500m -jar "%rmg%"\bin\RMG.jar condition.txt
```

Note: The name of the input file does not have to be `condition.txt`.

A log file containing the same information printed to the console, plus additional details, is automatically saved for you to `RMG.log` in the same directory as the condition file. Redirection of stdout to a file is no longer required.

9.3 Running RMG using Batch Scripts in Windows

As of version 3.2, several batch scripts have been provided to make running RMG on Windows more straightforward. These can be found along with the example condition files in the `examples` directory. There are currently three such scripts:

- `RMG.bat` - For running general RMG mechanism generation jobs
- `PopulateReactions.bat` - For generating possible reactions and kinetics of a set of species
- `ThermoDataEstimator.bat` - For estimating the thermodynamics of a set of species

The easiest way to use them is:

1. Create the condition or input file for the job you wish to run. For general RMG jobs this file should be named `condition.txt`. For other jobs this file should be named `input.txt`.
2. Copy the batch script corresponding to the job you wish to run to the folder containing the condition file. For best results, this folder should only contain the condition file and the batch script at this step. The batch script will happily overwrite existing files if they correspond to files created by RMG, including the results of a previous RMG run.
3. Double-click the batch script to start the job. A console window will appear. After a few seconds a line of text should appear that tells you the job has been started. If RMG is not setup correctly, an error message will appear instead telling you what needs to be done. As the job runs, output will be sent to the file `RMG.log` in the same directory as the condition file.
4. Close the console window when the job is finished. The batch script will print a line of text when the job is completed. This line of text does not reflect whether or not the job was successful; check the tail of `RMG.log` for that information.

Note: You can edit the batch script file using a program like Notepad if you want to change thing like the filenames or memory allocation.

9.4 Configuring paths using Environment Variables

You can control where a lot of output and temporary files are stored by manipulating environment variables before running RMG. This can be useful for running on networked clusters of computers, eg. for running

several jobs concurrently that can share the same quantum mechanics calculations results, or for saving your temporary scratch files on a local (fast) hard disk, and your final results on a shared network drive.

An example bash script provided in the 1,3-hexadiene example directory, demonstrates the variables and their default values:

The same approach should work for Windows.

You can also pass the variables when you call the program, eg:

```
$ RMG_QM_LIBRARY=$HOME/myQMlibrary java -Xmx500m -classpath $RMG/bin/RMG.jar RMG condition
```

CHAPTER

TEN

ANALYZING THE OUTPUT

10.1 RMG Output Files

RMG will generate several types of output files, each with a different purpose. These files will be located in the directory which contains the condition file. Generally, there are five outputs from RMG (six if pressure-dependence is turned on):

1. `output.txt`. This is a large file that shows the many steps RMG went through to build the model, including the addition of each new species, concentrations and fluxes at all of the reported integration time steps for each reaction system, and generally the details as RMG builds the mechanism. It is useful for debugging if a problem occurs, but other output files are better if all you are concerned about is the final model and its behavior. This file also lists the input file at the beginning.

2. `Final_Model.txt`. As its name would imply, this file contains the details of the final model created by RMG. For each Reaction System (combination of Temperature, Pressure, and Concentration) specified by the user, the following information is available:

   The concentration and mole fraction profiles of all "core" species, based on the time step information (the "TimeStep" or "Conversions" field) provided in the `condition.txt` file. This data can easily be pasted into a spreadsheeting program for analysis and plotting.

   The next set of information contained in the file is a list of all reactions included in the model, along with the modified Arrhenius parameters used for each reaction. For information on where these Arrhenius parameters came from (i.e. which reaction family), consult the `chem.inp` file. This set of data may also be pasted into a spreadsheeting program easily.

   Some of the reactions listed in the `Final_Model.txt` file may not have Arrhenius rate coefficient expressions; these reactions are pressure-dependent reactions generated by RMG. The pressure-dependent rate coefficient expression for each of these reactions is located in the `chem.inp` file.

   The next set of information contains the flux of each reaction, for each time step provided in the `condition.txt` file. A negative flux means the reverse reaction (as written) was dominant.

   The last set of information is the total running time. If sensitivity information was requested, it would also appear in this file.

3. `chem.inp` (located in `$RMG/chemkin`). This file contains the CHEMKIN model and associated thermodynamic data. The file lists the elements, species, thermodynamics data (in NASA-7 polynomial form), and reactions (without N2, Ar, He, or Ne) for the model.

If a species thermochemistry came from a Primary Thermo Library, a comment will precede the NASA-7 polynomial, stating which Primary Thermo Library it came from and the name of the species as identified in the Primary Thermo Library. For example:

```
!Primary Thermo Library: GRIMech3.0 (Species ID: s00009193)
CH4(2)                  C   1H   4          G   300.000  5000.000   995.043    1
 1.20933782E+00 1.07635862E-02-3.69789759E-06 5.80285027E-10-3.41693862E-14    2
-9.79206547E+03 1.24619300E+01 3.63765769E+00-2.58941555E-03 2.18452253E-05    3
-2.01605036E-08 6.08817863E-12-1.00975312E+04 1.65214016E+00                   4
```

This comment informs the user the species thermochemistry came from the species labeled "s00009193" in the GRIMech3.0 Primary Thermo Library.

A comment follows each reaction in the `chem.inp` file. The comments are usually of the following two forms:

```
C5H7J(13)+CH3J(14)=HXD13(1) 2.4874006e+15        -0.90000        0.25000         !R_Re
C5H7J(85)=C5H7J(13) 4.0800000e+05        1.91990         7.89680         !intra_H_migr
```

The first set of characters after the "!" reflect which reaction family the modified Arrhenius parameters came from. The next set of characters is either "exact" or "estimate"; this will be explained in further detail shortly. If the numbers are an "estimate", the word "Average:" will follow. The last set of characters are the functional groups RMG classified for this reaction.

If the kinetics are "exact", this means RMG has a number in its database for the set of functional groups listed. In the previous example, RMG found kinetics in the "intra_H_migration" reaction family for the set of nodes: {R5H_DSMS, Cd_rad_out_singleH, Cs_H_out_2H}.

If the kinetics are an "estimate", RMG did not find numbers for the set of functional groups listed. Thus, RMG could not find kinetics in the "R_Recombination" reaction family for the set of nodes: {C_rad/H2/Cd, C_methyl}. So, an averaging scheme was performed to generate the reported modified Arrhenius parameters.

The previous two examples were generated by setting the "Verbose" field in the `condition.txt` file "off". If the user sets the "Verbose" field "on":

```
C5H7J(13)+CH3J(14)=HXD13(1) 2.4874006e+15        -0.90000        0.25000         !R_Re
```

If you wish to immediately run the generated `chem.inp` file in CHEMKIN, please set the "Verbose" field to off; some of the "Average of: ..." comments span too many characters for the CHEMKIN Pre-Processor to handle.

Lastly, if the reaction (and thus, kinetics) came from a Reaction Library or Seed Mechanism, the comments will reflect this. If the reaction was generated by RMG, but the kientics were taken from a Primary Kinetics Library, that information will also be reported here.

4. `RMG_Dictionary.txt`. This file lists all the species by name that are used in the model, as well as their adjacency list; if the *InChI generation feature* (page 36) is turned on, the file will also contain the InChI representation for each molecule. This file allows the user to see a graphical representation of all species in the model. Typically the user will copy this file to an appropriate directory so that it can be read by the RMGVE (See Section *Viewing the Species in the RMG Viewer/Editor (RMGVE)* (page 85) for further details).

5. `tran.dat` (located in `$RMG/chemkin`). This file contains the estimated transport properties for all "core" species in the mechanism. The source of each species' transport data is listed at the end of the line, for example:

```
HXD13(1)            2    356.098      5.979      0.000      0.000      1.000 ! LJ parameters
CH4(2)              2    141.400      3.746      0.000      2.600     13.000 ! Primary Trans
```

   The first species' properties were estimated by RMG. The group-additivity estimated critical properties and boiling point are listed first, followed by the node names that RMG matched in performing the group-additivity method. The complete set of transport nodes may be found in `$RMG/databases/RMG_database/transport_groups/`.

   The second species' properties were read-in from a "Primary Transport Library" supplied by the user, in this case, the library named "GRIMech3.0". The name of the species RMG matched is listed at the end of the comments, in this case, CH4.

6. `tableOfRateCoeffs.txt` (located in `$RMG/chemkin`). If pressure-dependence was turned on, an additional file will be stored in `$RMG/chemkin`. This file contains the computed k(T,P) for all reactions present in the mechanism, for each temperature and pressure requested by the user in the `condition.txt` file. These rate coefficients were used to compute the reported Chebyshev or PLOG pressure-dependent rate coefficients reported in the `chem.inp` file.

7. `SpeciesProfiles_N.txt` (located in `$RMG_JOB_SCRATCH/ODESolver`). This is a tab-separated text file (suitable for opening in a spreadsheet like Excel) containing the concentration of each species (moles per cm^3) as a function of time (seconds). There is an entry for every timestep of the ODE solver. This is particularly useful when running with AUTO timesteps in the *Dynamic Simulator* (page 42). The number at the end of the filename indicates which reaction system it corresponds to (e.g. if multiple reaction systems are set up at different *initial conditions* (page 35)). Because the file is replaced every time the ODE solver runs, at the end of running RMG it will contain the results from the most recently run simulations.

8. `restartConditionFile.txt`. This file is a modified condition file which can be used for restarting a RMG job. The original condition file is appended with all the species in the current core (not contained in the seed) with zero concentration. All other settings/options in the original condition file are preserved. Running an RMG job using this condition file causes the original model to be recovered after the first time step.

10.2 Viewing the Species in the RMG Viewer/Editor (RMGVE)

When RMG has finished, it will create a file `RMG_Dictionary.txt` in the same directory as the `condition.txt` file. This file contains a list of all the core species (usually the edge species are too numerous to be of use); the core species can be viewed easily using the RMGVE. To view the core species, rename the first part of the file (this step isn't strictly necessary, but it is convenient and a good practice). The file must still end in "`_Dictionary.txt`" (e.g. `newname_Dictionary.txt`) for the RMGVE to read it. Next, copy this file to the directory `$RMG/databases/RMG_database/thermo_groups`. Once the file is in the `thermo_groups` directory, double-click on the `RMGVE 20080101.bat` file to launch the program. Please note that you must close and relaunch the program every time you add a new file to the `thermo_groups` directory. Once the RMGVE has launched, double-click on the folder `Thermo`. Your new file should appear in this directory (e.g. newname without the "`_Dictionary.txt`"). Push the

"Read family" button and a new window should appear with a list of names. Highlight any name and click "View" to see its structure.

CHAPTER

ELEVEN

MODIFYING THE RMG DATABASES

11.1 Editing the Thermodynamic Database

As mentioned in Section *Primary Thermo Library* (page 33), it is possible to override the default thermodynamic values and substitute your own thermodynamic data.

11.1.1 Basic Structure of the Thermo Database

The thermodynamics database consists of three sections, each of which is an ASCII file that can be edited to alter the information. This description applies to non-radical groups. There are other tree/library/dictionary files for radical groups, ring corrections, and other corrections. The nomenclature is different, but they would be edited in a similar way. The files of interest are located in the directory: `$RMG/databases/RMG_database/thermo_groups/`. The three files are described below.

Dictionary File

`Group_Dictionary.txt` contains the name and adjacency list (structure) for all of the nodes contained within the thermo tree. The nomenclature within all three files must be identical. The asterisk "*" denotes the central atom in the group for which the group value is defined. The format for each line in the adjacency list is: atom #, * if central atom, element, # of radicals (0, 1, 2) on element, bonding:

```
Cb-(Cd-Cdd-Cd)
1 * Cb   0  {2,S}
2   Cd   0  {1,S} {3,D}
3   Cdd  0  {2,D} {4,D}
4   C    0  {3,D}
```

Tree File

`Group_Tree.txt` defines the tree structure of the database. The nodes at any particular level are defined by including the text "`Lx:`" prior to the name of the group at that node, where "`x`" is a number corresponding to the level in the tree. A sample is given below:

```
L0: R
  L1: C
    L2: Cbf
      L3: Cbf-CbCbCbf
    L2: Cb
      L3: Cb-H
      L3: Cb-Os
      L3: Cb-C
        L4: Cb-Cs
        L4: Cb-Cd
          L5: Cb-(Cd-Od) // Cb-CO
          L5: Cb-(Cd-Cd)
            L6: Cb-(Cd-Cd) // Cb-Cd
        L4: Cb-Cb
    L2: Ct
```

Note that the indentation is not necessary because it is the "`Lx:`" that the software reads, but is very helpful in making these files human readable.

Library File

`Group_Library.txt` is the archive of the actual data associated with a given group in the tree and dictionary. There are 15 (space or tab separated) fields in the thermo library to describe the group. The units for enthalpy are kcal/mole, and the units for entropy and heat capacity (Cp) are cal/mole-K. The columns are described in the table *"Library File Definitions"* (page 88).

Table 11.1: Table: Library File Definitions

Column	What it contains
1	A unique number; this does not correspond to any other part of the thermo database, but numbering sequentially is most logical
2	Group name; same as in tree and dictionary
3-4	Enthalpy and Entropy at 298K
5-11	Cp at T = 300, 400, 500, 600, 800, 1000, and 1500K
12-13	dH and dS: absolute uncertainties in the enthalpy/entropy estimates
14	dCp: absolute uncertainty in the Cp estimate (no temperature consideration)
15	Comments Section: usually citing the source of the data or comments on the reliability

A sample entry can be found in Section *Editing the Data for an Existing Thermo Functional Group* (page 92).

11.1.2 Thermo Database and Adjacency List Notation

In general, the thermo database uses what are known as function group elements. Function group elements serve to define the atom and its bonding environment. These definitions serve to simplify the adjacency lists of groups and allow for more general descriptions of groups. The notation used in the database is shown below. New functional group elements cannot be added to RMG in a simple way, as they must be hard-coded into the RMG software with the appropriate properties.

If you examine the file `Group_Dictionary.txt`, you will see that these groups are used extensively, even more so than the actual atoms C, H, or O. You will also see that groups can be defined using a bracketed notation, which simply means that either atom/functional group element will generate this node. For example:

```
Cb-Cd
1 * Cb        0   {2,S}
2   {Cd,CO}   0   {1,S}
```

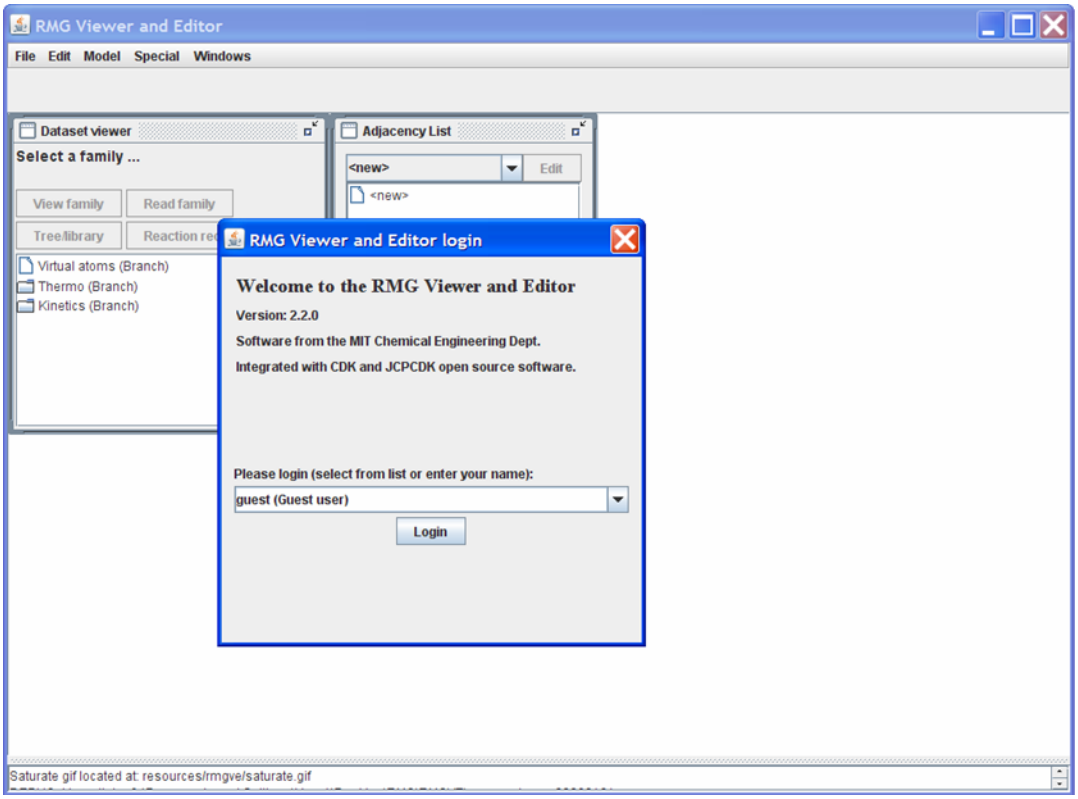
In this example, the second "element" is actually either Cd or CO. The reason that either Cd or CO will generate the same node is because both fall under the more general definition, which is a carbon atom with one double and two single bonds. This can also be useful if there is a secondary effect and all that matters is that there is a π-bond present, but the fact that it is C=O or C=C does not matter. This is not used very much in the thermo database, but occurs much more in the kinetics databases where radical delocalization can play a major role in determining the rate coefficient of a reaction. A table of the possible functional groups can be seen in the table *"Functional Group Elements"* (page 89).

Table 11.2: Table: Functional Group Elements

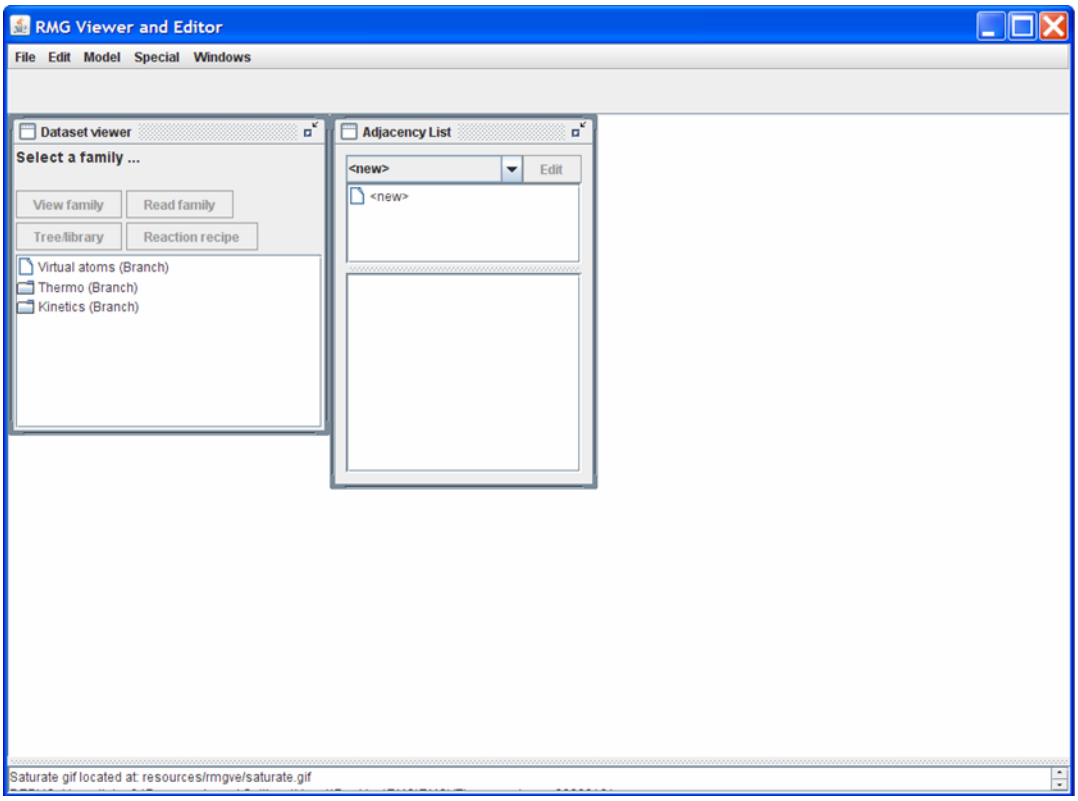
Symbol	Definition
Cs	Carbon bonded to four single bonds
Cd	Carbon bonded to a double bond and two single bonds. (The other end of the double bond is carbon)
Cdd	Carbon bonded to two double bonds
Ct	Carbon bonded to a triple bond and single bond
Cb	Carbon bonded to two benzene bonds and a single bond. (The carbon belongs to only one benzene ring)
Cbf	Carbon bonded to three benzene bonds (the carbon belongs to two or three benzene rings)
CO	Carbon bonded to a double bond and two single bonds. (The other end of the double bond is oxygen)
Os	Oxygen bonded to two single bonds
Od	Oxygen bonded to a double bond
Oa	Oxygen triplet
R	Any atom
R!H	Any non-hydrogen atom

11.1.3 Viewing a Thermo Functional Group in the RMGVE

Open the RMG Viewer & Editor (RMGVE) by running the `RMGVE 20080101.bat` file. If your RMG database is located in the `$RMG\databases\RMG_database` folder, the following login screen should appear.



The login name is used to document who made what changes in the RMG libraries. After entering your name, press the "Login" button to reveal the RMGVE.

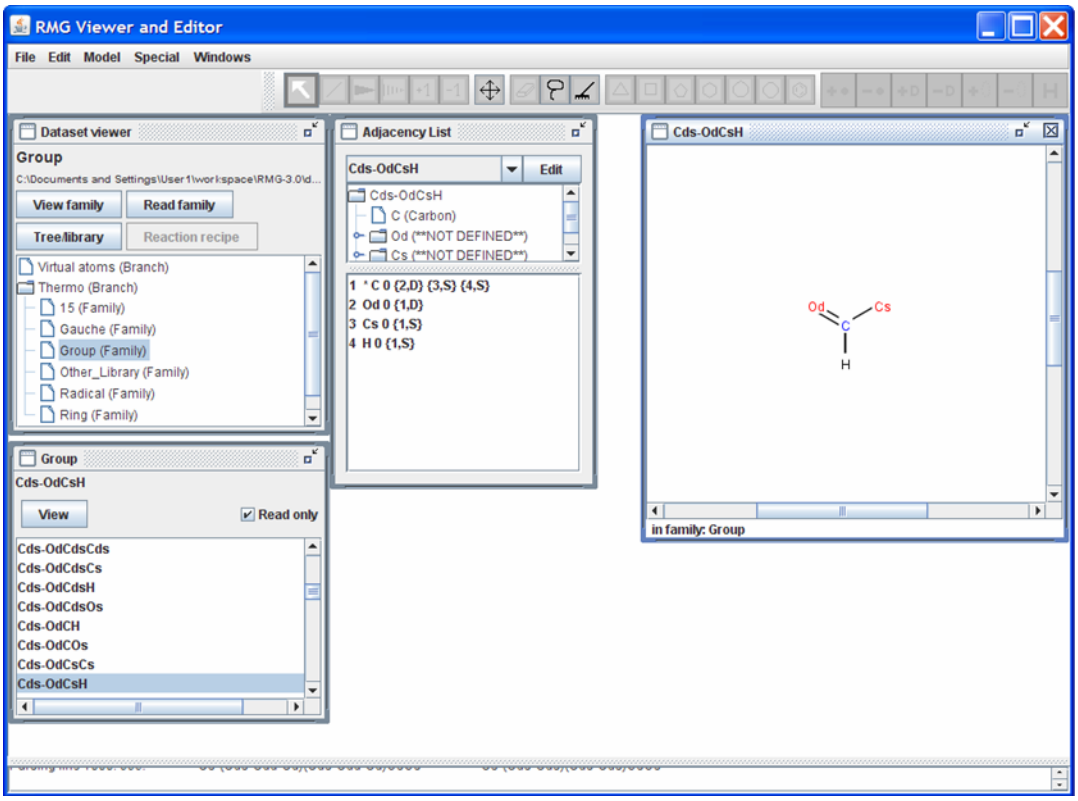


The RMGVE home screen initially has two windows open: the "Dataset viewer" and the "Adjacency List." In the "Dataset viewer" window, double-click on the "Thermo (Branch)" folder to show the different Thermo Families present in the `RMG_Dictionary` folder. To load one of the Thermo Families, either:

- double-click on one of the names OR
- highlight one of the names and then push the "Read family" button near the top of the "Dataset viewer" window

Note: Loading a family could take a few minutes, depending on how large the family is.

Once the family is loaded, a new window will pop up. The name of the window corresponds to the name of the Thermo family that was read. In the screen shot shown below, we have chosen to load the "Group" family. Use the scroll bar to see the list of functional groups contained in the "Group" family. To visualize what the different functional groups look like, highlight one of the names and then push the "View" button near the top of the "Group" window. For instance, if we select the functional group name "Cds-OdCsH," the RMGVE should look as follows.



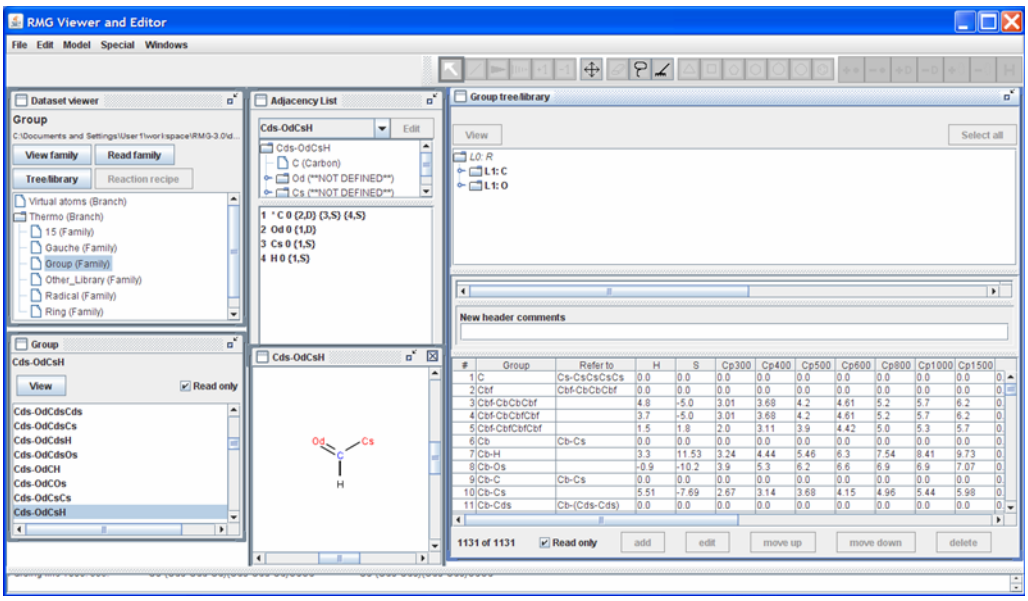
Notice that the "Adjacency List" window is no longer blank but now contains the adjacency list for the functional group "Cds-OdCsH". Furthermore, a window entitled "Cds-OdCsH" opens which contains a visualization of the functional group.

Note: Notice also that a toolbar appears above the "Dataset viewer," "Adjacency List," and "Cds-OdCsH" windows. This toolbar may be used to edit the molecule in the "Cds-OdCsH" window. For instance, click on the icon with the four arrows (pointing up, right, down, and left). Now click on one of the atoms in the "Cds-OdCsH" window and drag it to another location within the window. To restore (clean) the structure of the functional group, click on the icon containing the picture of a rake.

11.1.4 Editing the Data for an Existing Thermo Functional Group

A leaf in the thermochemistry tree may be **edited** using the RMG Viewer & Editor (RMGVE). For instructions on how to add a leaf to a thermochemistry tree, please refer to the section *Adding Additional Nodes to the Thermo Database* (page 97).

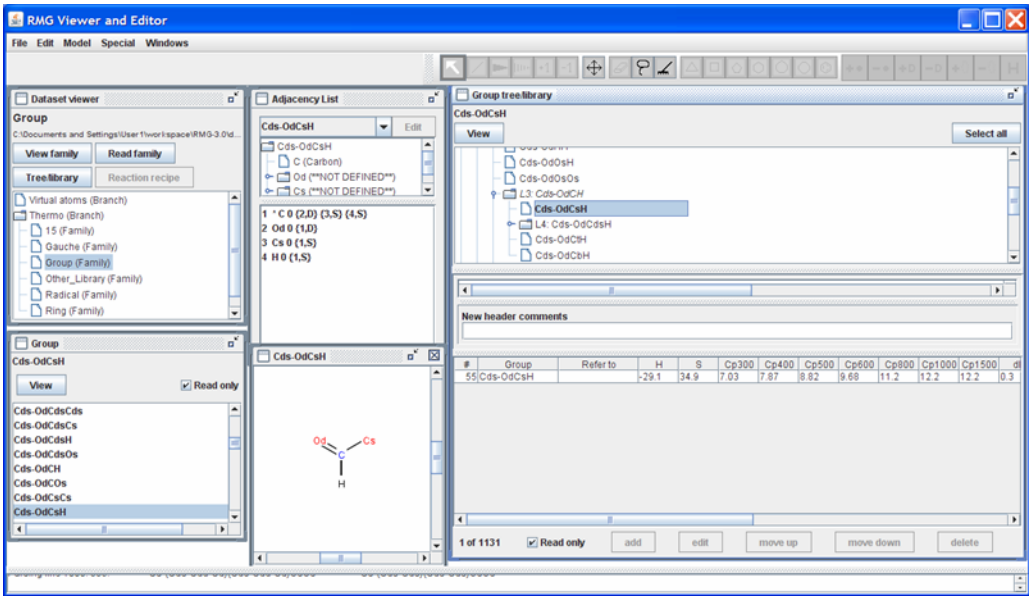
Returning to the "Dataset viewer" window, notice the "Tree/library" button has been enabled. Click on this button to show the data for the highlighted thermo family. After some re-arranging of the windows, the screen should look as follows.



The top half of the new "Group tree/library" window contains the "Group" thermo family tree; the bottom half contains the "Group" thermo family library. Suppose we want to change the thermo value for the "Cds-OdCsH" functional group. To do so, we can navigate the tree in the top half of the "Group tree/library" window by opening (double-clicking) the folders corresponding to the "Cds-OdCsH" functional group:

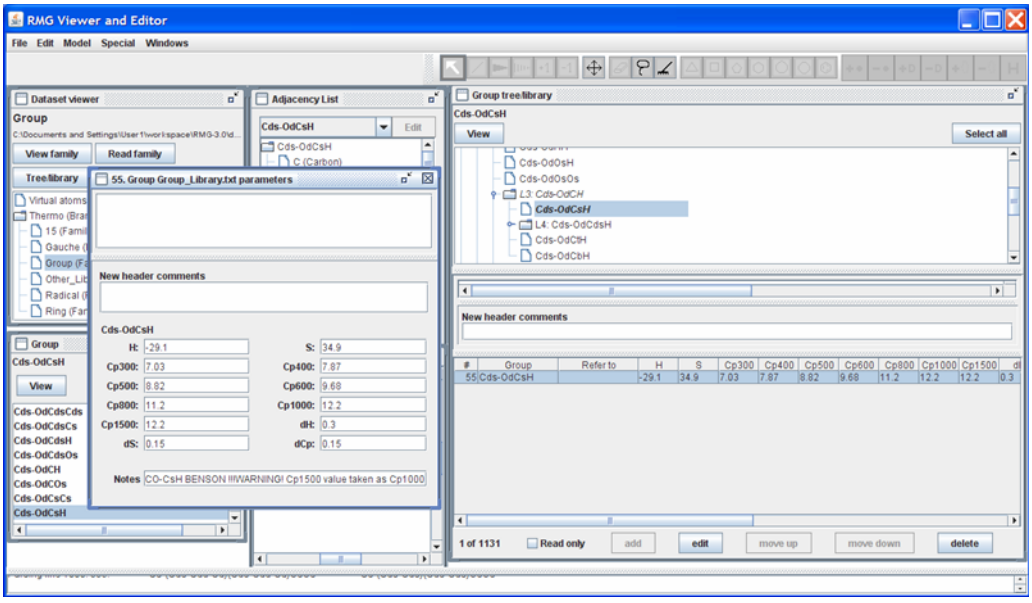
- L1: C
- L2: Cds
- L3: Cds-OdCH
- Cd-OdCsH

After navigating the tree, the RMGVE should look like the following snapshot.



Notice that RMG has thermodynamic data for this functional group. If you scroll to the right (using the scroll bar near the bottom of the "Group tree/library" window) far enough, you will see the "Notes" column. This column is used to document where the data came from. In this case, the thermodynamic data comes from the Benson defined group O=CH-Cs. Furthermore, it has been noted that the Cp1500 was assumed to be the Cp1000 value.

Suppose we had a better estimate for the Cp1500 value (e.g. 12.5 cal/mol/K). To edit this leaf, uncheck the "Read only" field, highlight the row, and push the "Edit" button; the screen should look as follows.



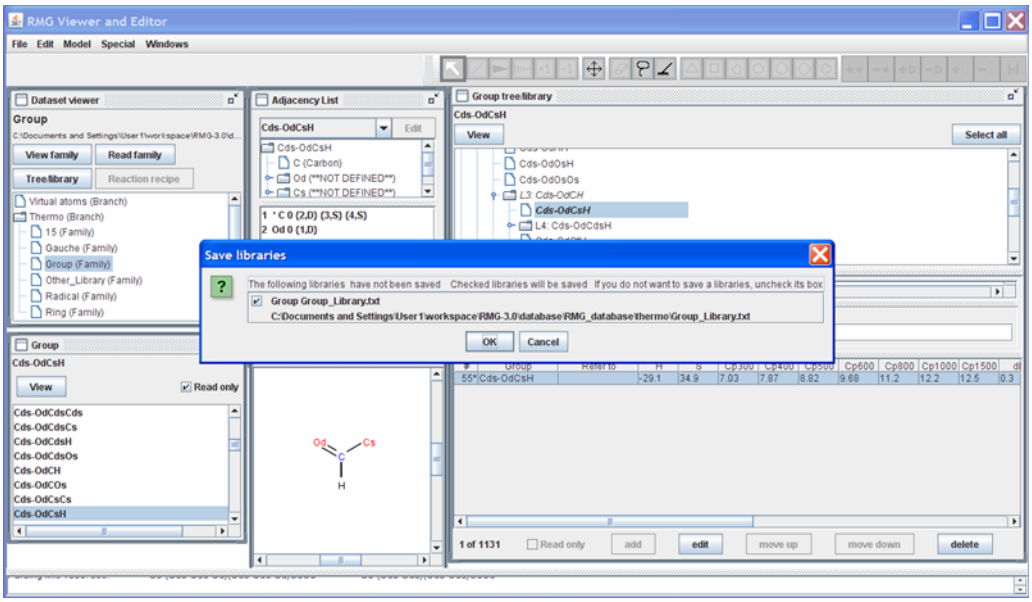
Enter the new thermochemical data (Cp1500 = 12.5) in the appropriate field. The field "Notes" will be added to the thermochemical data for the "Cds-OdCsH" functional group at the end of the line (after the 12 data entries). The field "New header comments" will be placed above the thermochemical data for the "Cds-OdCsH" functional group. Upon changing any of the thermochemical data fields, a line in the "New header comments" will automatically appear, containing your login name and the current date and time. The

box immediately above the "New header comments" field shows how the RMGVE will edit the current data in the `Group_Dictionary.txt` file. Once all data has been entered, close the window.

Returning to the "Group tree/library" window, notice the highlighted row. The thermochemical data entries have been updated and a star (*) has been placed next to the "55" value in the first column. The star (*) reflects that a change has been made.

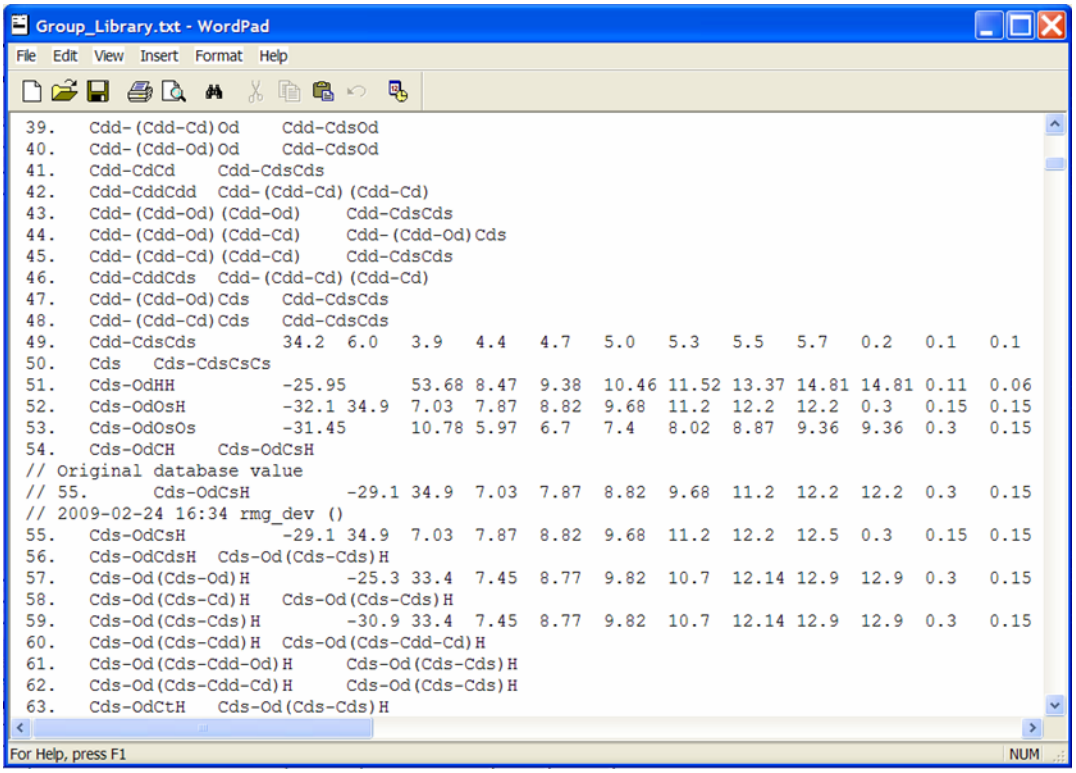
When you have completed making changes to the thermochemistry database, close the RMGVE. Upon doing so, you will be asked which changes you would like to accept and save to the `RMG_database` folder. Before pushing the "OK" button, ensure that all of the changes you want to be saved have a checkmark next to them. You will receive a message stating whether the edits were saved successfully or not. Pushing "OK" in this window will close the RMGVE.

Note: Notice the RMGVE directs you to the location of the file that is about to be changed.



Viewing the changed thermochemical data

To view the thermochemical data that was just edited, open the file from the previous screenshot: `$RMG/databases/RMG_database/thermo_groups/Group_Library.txt`. Scrolling down to the entry #55, we see the old and new thermochemical data for the functional group "Cds-OdCsH".



Rather than explicitly entering data for each leaf, one may also refer one functional group to another, if you believe they should be the same. This is done by putting the name of the group that has the same thermo parameters in column 3 and leaving columns 4-15 blank. An example of this is entry #56 where we see that this leaf does not contain thermodynamic data but rather points to the functional group "Cds-Od(Cds-Cds)H," entry #59. Effectively, if RMG encounters the group and finds the name of another group instead of numerical values, it will assign the current group the same values that occur in the referred group:

```
56.     Cds-OdCdsH      Cds-Od(Cds-Cds)H
59.     Cds-Od(Cds-Cds)H                -30.9   33.4    7.45    8.77    etc.
```

The above example would assume that the group values for #56 are identical to those of #59.

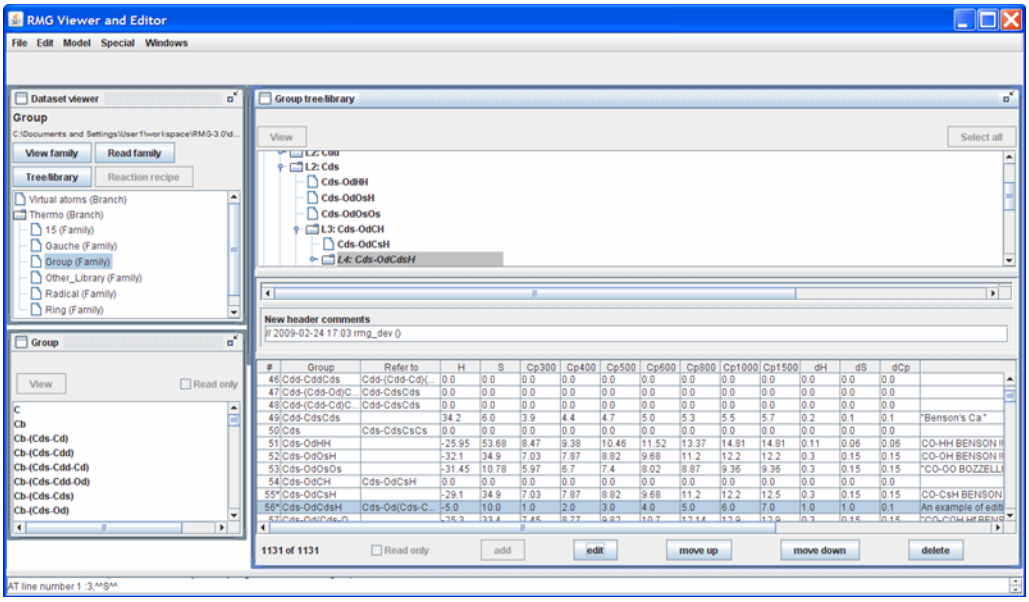
This referencing does not need to be done if the group to which you want to refer lies directly above the current group in the tree, because if the tree does not have a certain node defined it will look back up the tree to find the nearest relative and use those values:

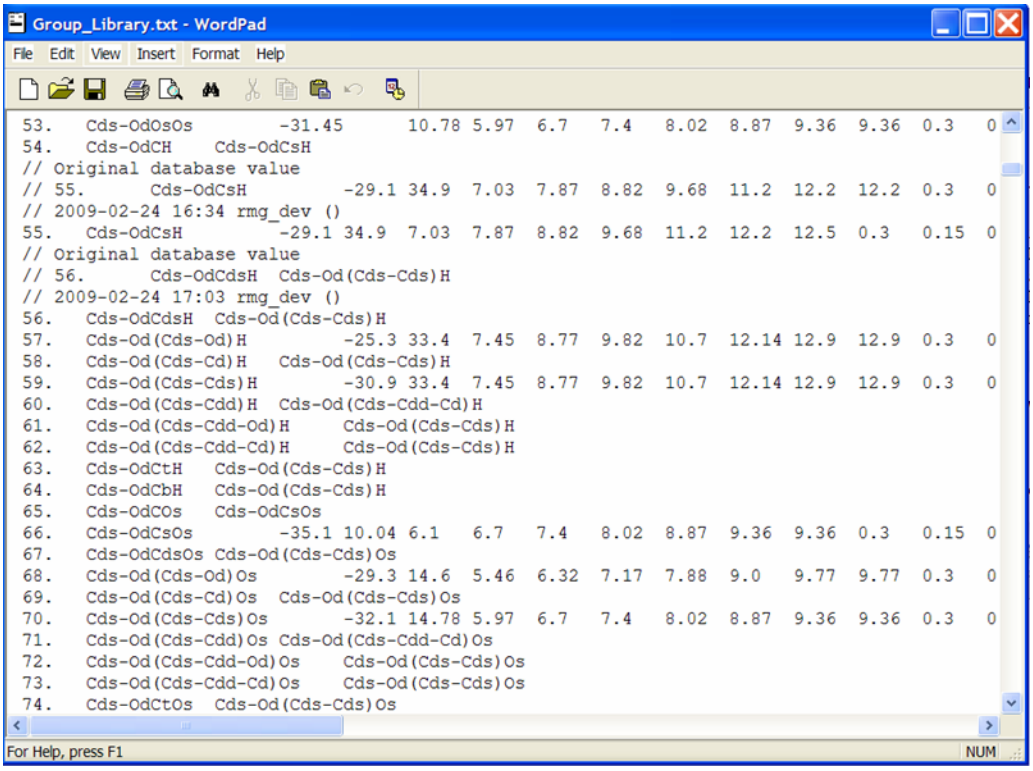
```
253   Cb-(Os-(Os-Cs))   -2.5   -8.5   etc...
254   Cb-(Os-(Os-H))    Cb-(Os-(Os-Cs))
255   Cb-(Os-(Os-(Cs-OsHH)))   Cb-(Os-(Os-Cs))
```

In this case, the referring of #255 back to #253 is unnecessary because RMG would refer back to #253 by default if it did not find any values for #255. This redundancy will not create any problems within RMG, but it is simply unnecessary.

Warning: The RMGVE can only edit leafs which already have thermochemical data stored for them. In particular:

- The RMGVE does not allow a user to change the "Refer to" field. This change must be performed manually.
- The `Group_Library.txt` file will not recognize changes made in the RMGVE to leafs that "Refer to" another species. For example, had we entered thermochemical data for entry #56, the RMGVE will show the updates to the H, S, and Cp values in the "Group tree/library" window but will also still show the "Refer to" functional group. After closing the RMGVE and confirming the change to the database, if one opened the `Group_Library.txt` file and looked at entry #56, you will notice the leaf's thermodynamic values were not updated.





Please be aware that version control may be a significant issue with the databases and should be addressed early to ensure consistency within the group.

11.1.5 Adding Additional Nodes to the Thermo Database

This task is more complicated than the previous example because it involves altering all three database files in a consistent manner, without the help of a Graphical User Interface. It is worth noting that the order in which species are added to the dictionary (`_Dictionary.txt` file) and library (`_Library.txt` file) does not matter; however, the position where the new group is added to the tree (`_Tree.txt` file) is of the utmost importance. It is useful to create the tree and dictionary with items in the same order. This ordering will facilitate cross checking and debugging if needed. Since all files must use the same nomenclature, the ability to search may make consistent ordering unnecessary. The procedure is described below.

1. Find the appropriate location to place the new group and ensure that the nomenclature is unambiguous and unique. Placing the group in the incorrect location could cause incorrect estimates to be made when the tree is being searched. Using the RMGVE to navigate the tree structure is a useful way to find the location for your new group.

2. Add the line of text to the `Group_Tree.txt` file in the following form, where "`x`" is the appropriate level. The following example is for a O-O-H off of a benzene ring.:

   ```
   Lx: Cb-(Os-(Os-H))
   ```

3. Using the same name as in the tree, append the file `Group_Dictionary.txt` to define the structure of the group and its atom center (denoted by the asterisk):

```
Cb-(Os-(Os-H))
1* Cb 0 {2,S}
2  O 0 {1,S} {3,S}
3  O 0 {2,S} {4,S}
4  H 0 {3,S}
```

4. Add the new group to the `Group_Library.txt` file using the same nomenclature and whatever thermo data you have for the group. The format was shown in the previous section.:

```
2374 Cb-(Os-(Os-H)) 3.5 10.0 ... "I added this b/c ..."
```

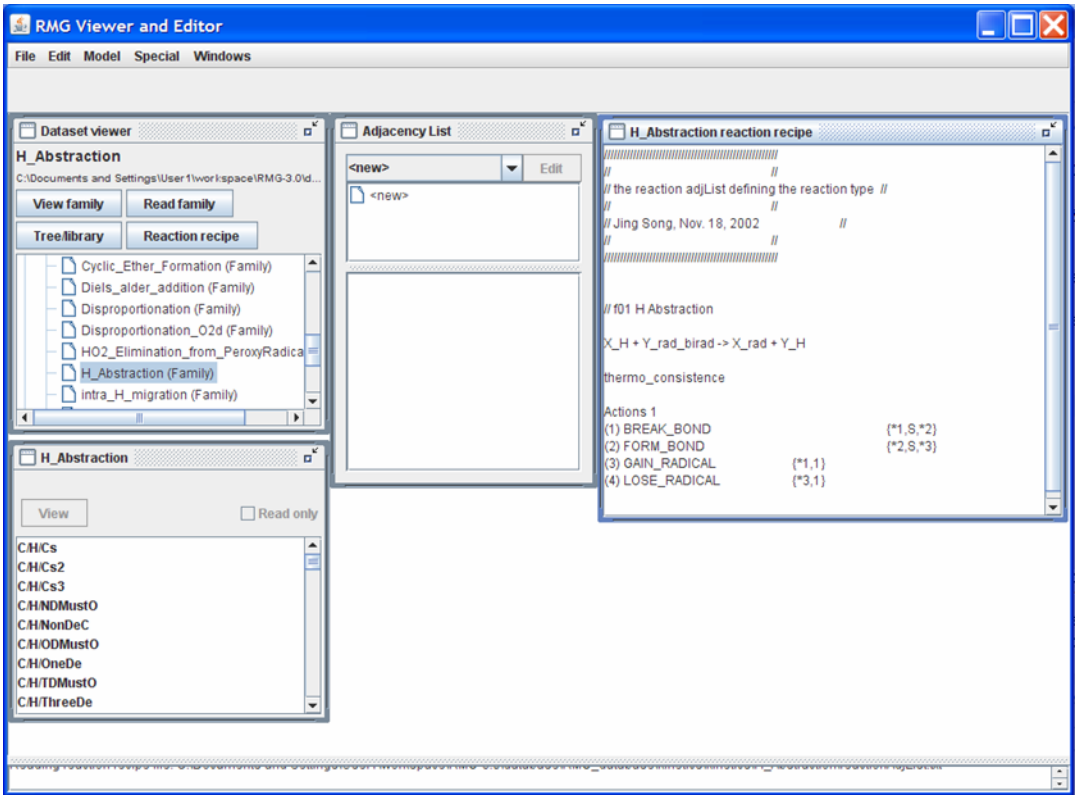
11.2 Editing the Kinetics Database

RMG's kinetics database may also be viewed and edited using the RMGVE. From the RMGVE's home screen, double-click on the "Kinetics (Branch)" folder in the "Dataset viewer" window and then double-click on the "kinetics (Family)" folder. Scroll down to the family entitled "H_Abstraction." Highlight this family and push the "Read family" button. All 4 buttons in the "Dataset viewer" window should now be active.

1. "Read family": This button reads in the functional groups pertaining to the highlighted library and displays them in a new window; the name of the new window corresponds to the thermo/kinetic family name. (Option available for both thermo and kinetics).
2. "Tree/library": This button opens a window that displays the thermochemical data for the highlighted library. (Option available for both thermo and kinetics).
3. "Reaction recipe": This button opens a window that displays the definition of the reaction family. (Option available ONLY for kinetics).
4. "View family":

11.2.1 Viewing a Reaction Recipe in the RMGVE

Push the "Reaction recipe" button. Your screen should look like the following snapshot (after rearranging and resizing the windows)

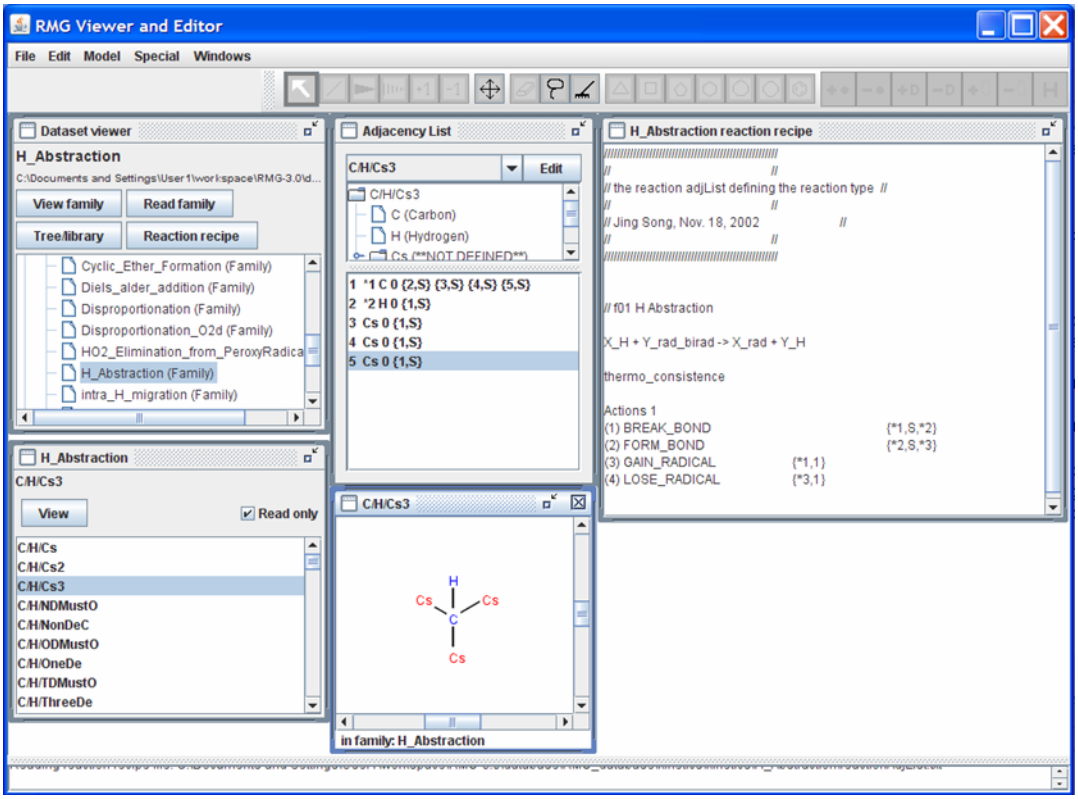


Looking at the "H_Abstraction reaction recipe" window: The first non-commented line shows the generic reaction: `X_H + Y_rad_birad -> X_rad + Y_H`. The next uncommented line informs RMG how to handle the reverse reaction. Some common occurences you will find include:

- "thermo_consistence": RMG may use the forward reaction rate coefficients stored in the <rxnFamilyName> folder. This nomenclature is only used when the forward and reverse reaction templates are the same (e.g. H_Abstraction and intra_H_migration)
- "none": RMG assumes the reaction is irreversible
- "(f_): <rxnFamilyName>": RMG will use the numbers stored in the rateLibrary.txt file for the forward reaction, and will use microscopic reversibility to compute the reverse reaction rate coefficient

The next set of uncommented lines describes the changes in the reacting molecules', "X_H" and "Y_rad_birad", connectivity graphs to form the products, "X_rad" and "Y_H". For the "H_Abstraction" reaction family, a single bond ("S") is broken between atoms "1" and "2" and a single bond ("S") is formed between atoms "2" and "3". Furthermore, atom "1" gains a radical while atom "3" loses a radical.

For a better understanding of what the atom numbering means, click on one of the species in the "H_Abstraction" window. If you click on the species "C/H/Cs3", your screen should look like the following snapshot



Looking at "C/H/Cs3" and "Adjacency List" windows reveals the identity of atoms "1" and "2". Looking at the "Adjacency List" window, notice the atoms that have the $\star$# notation between the counting index and the elemental symbol. The number after the $\star$ corresponds to the number in the reaction recipe. The atoms of interest are also shown in blue in the "C/H/Cs3" window.

11.2.2 Editing a Reaction Family using the RMGVE

Returning to the "Dataset viewer" window, click the "Tree/library" button. The bottom-half of this window contains the following kinetic information:

1. Groups: The columns entitled "Group 1", "Group 2", etc. represent the structure of the reactant(s).
2. Temp: This column contains the Temperature range (in units of Kelvin) over which the kinetic data is valid.
3. A, n, a, E0: These columns contain the modified Arrhenius parameters (A, n, E0) and the Evans-Polanyi coefficient (a). "A" has units of "mol,cm3,s" and "E" has units of "kcal/mol".:

   ```
   The Evans-Polanyi principle states that for a series of closely-related reactions,
   a linear relationship between the activation energy (Ea) and the enthalpy of
   reaction (delHr) is sometimes observed and can be expressed as:

   Ea = E0 + a*delHr

   where E0 and a are empirically-derived constants.
   ```

4. dA, dn, da, dE0: These columns contain the error for the Arrhenius parameters and Evans-Polanyi

coefficient. If the number is reported with a star (*) in front of it, this represents a multiplicative error; else, the error is additive.

5. Rank: This column contains the confidence the RMG developers have in the kinetic data on a scale of 1 to 5, with 1 being "very confident" and 5 being "rough estimate".

6. Notes: This column contains notes on where the kinetic data came from. The reaction library is not as well-documented as the thermo library.

Suppose we want to add a reaction rate coefficient to RMG for the following reaction:

```
H3C-CH2-CH3 + *CH3 = H3C-*CH-CH3 + CH4
```

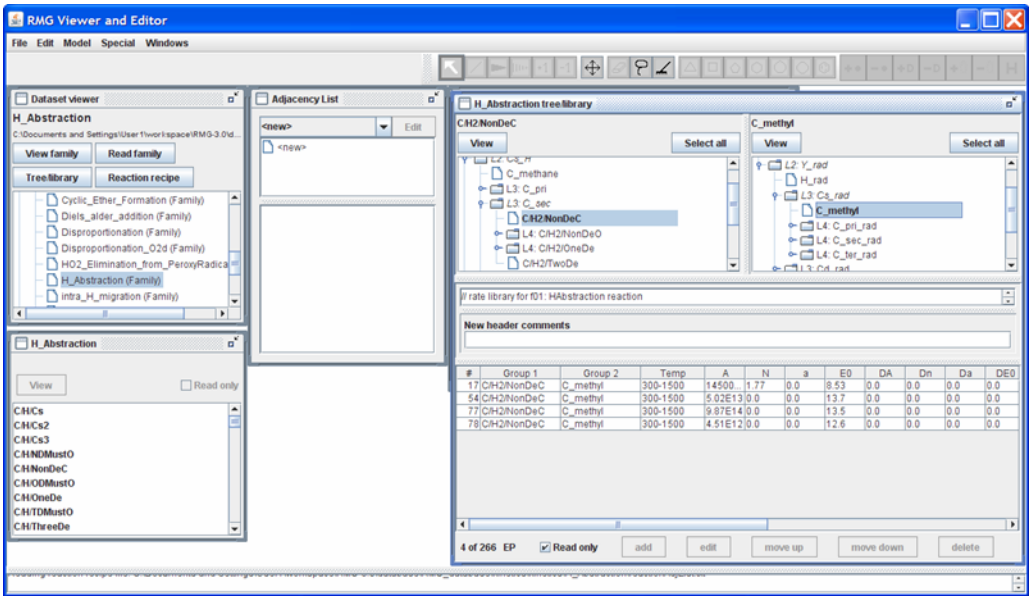
The first step would be to identify which of RMG's reaction templates the reaction of interest corresponds to. For our case, the reaction template we want is "H_Abstraction". The next step is to find the pair of nodes in the tree that correspond to the reaction of interest. Looking at the tree, we need to identify a "X_H" and "Y_rad_birad". For the example, "X" is the central carbon of propane and "Y" is the methyl radical. Traversing down the "X_H" tree leads us to the following node:

```
L1: X_H
L2: Cs_H
L3: C_sec
C/H2/NonDeC
```

If you are ever uncertain of the structure of one of the nodes, highlight it and push the "View" button to open a window with a 2-d graphical depiction of the species. Traversing the "Y_rad_birad" tree leads us to:

```
L1: Y_rad_birad
L2: Y_rad
L3: Cs_rad
C_methyl
```

Your screen should look something like the following

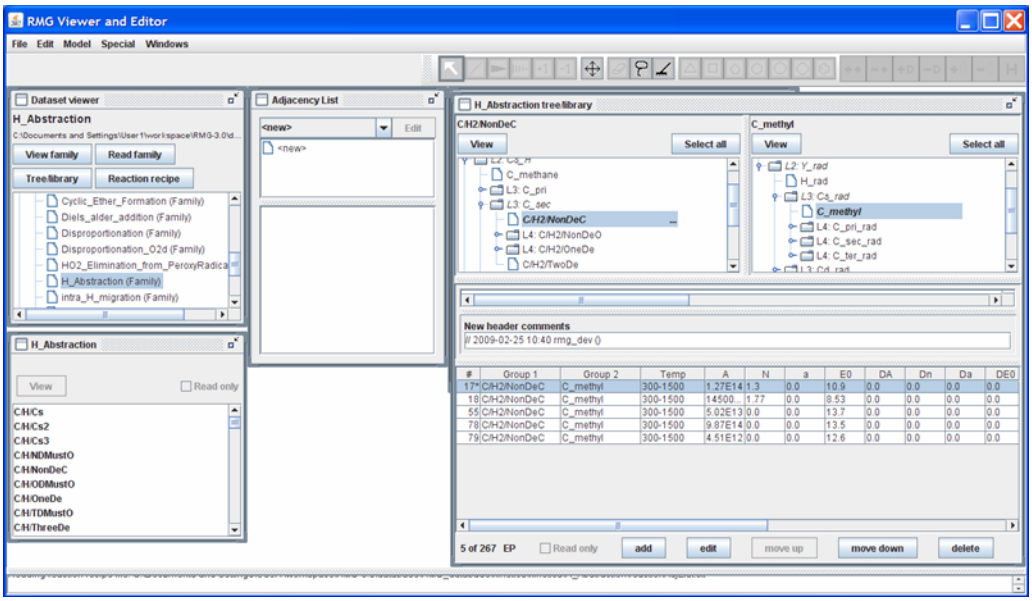


Notice that RMG contains 4 different sets of kinetic data for this reaction. There are two explanation for this:

1. The reaction rate coefficients all pertain to the same reaction: The different values could pertain to experimental vs. theoretical values or differing basis sets in the theoretical calculations (e.g. using an ab initio quantum chemistry sofware package)
2. The reaction rate coefficients pertain to different reactions: Notice that the "X_H" leaf is only defined as "C/H2/NonDeC". Thus, one of the reaction rates could correspond to the methyl radical abstracting a Hydrogen atom from:
   - propane to form the iso-propyl radical
   - butane to form the sec-butyl radical
   - ...

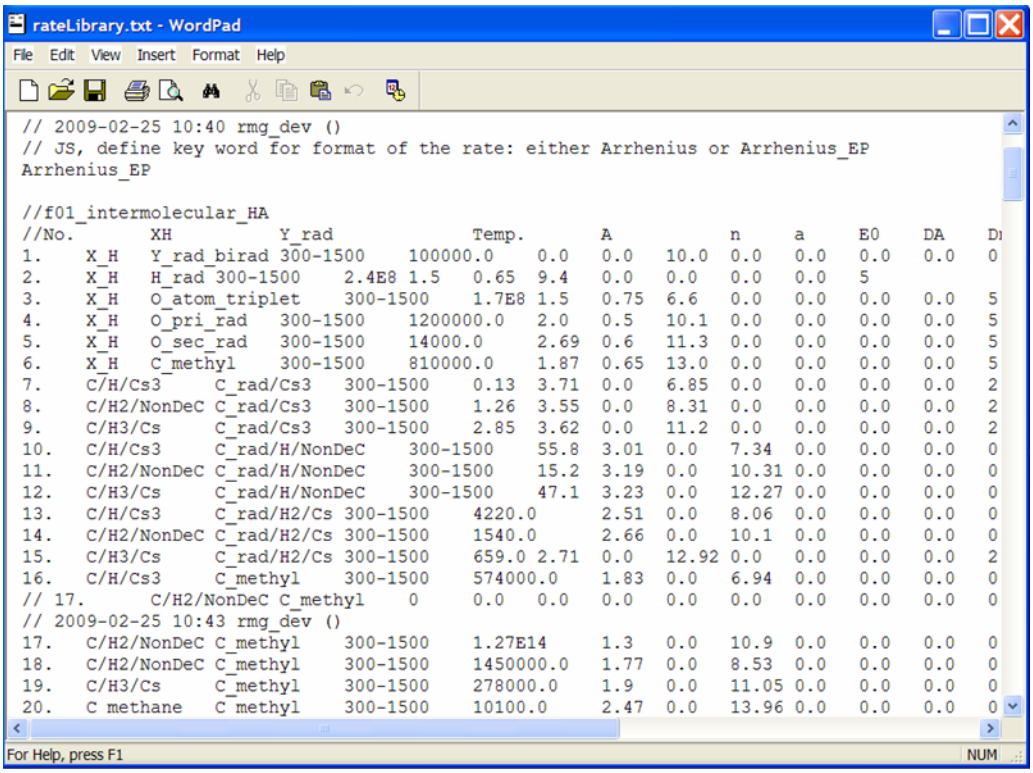
For RMG's reaction library, the order in which the reactions appear in the `rateLibrary.txt` file is significant. In the instance when multiple rate coefficients are found for the same sets of nodes, RMG will first eliminate those rate coefficients whose valid temperature range does not contain the simulated temperature. Next, RMG will find the rate coefficient with the lowest rank. In the event multiple kinetic rates have the same Rank and their valid temperature range contains the temperature of interest, RMG will use the kinetics from whichever instance was read in first (i.e. is closer to the top of the file). If you click on one of the 4 reaction rates, you'll notice that the RMGVE gives you the option of edit[ing] or delete[ing] the entry (as it did for the Thermo library) but it also gives you the option to "move up" or "move down" the reaction rate.

We can also add a new entry, by pushing the "add" button. For example, let's suppose we have the following modified Arrhenius parameters for the reaction mentioned above: A = 1.27E14 cm3/mol/s, n = 1.3, E = 10.9 kcal/mol over the temperature range 300-1500K. Push the "add" button, then push the "edit" button, and then fill in the popup window with the kinetic data. After closing the popup window, your screen should look like this



Close the RMGVE. A popup window will ask if you want to save the changes you have made. If you leave the box checked and then open the

`$RMG/databases/RMG_database/kinetics_groups/H_Abstraction/rateLibrary.txt` file, you should see that the file has been updated.



11.3 Creating a Primary Kinetic Library / Reaction Library / Seed Mechanism

Reaction libraries and seed mechanisms are all stored in `$RMG/databases/RMG_database/kinetics_libraries`

A primary kinetic library / reaction library / seed mechanism may consist of three files:

- `species.txt`
- `reactions.txt`
- `pdepreactions.txt`

The `species.txt` file is **required**. All other files are optional, and if present, must include the Unit declaration and Reaction headings.

1. The `species.txt` file is a series of molecular names and connectivity lists, analogous to the formats used in the input file `condition.txt`. All species present in the two remaining files must be given a structure in `species.txt`. The structure should have the same format as the adjacency list shown in *Creating a condition file* (page 31). Please note that the names that are used in the reaction library will be used throughout the mechanism. In this manner, the user can adopt a preferred nomenclature for individual species.

2. The `reactions.txt` file defines the standard high-pressure limit reaction kinetics. The file has the structure shown in the following sample. Comments are denoted with "`//`" and are ignored by the

RMG parser.:

```
// Define the units
// Units allowed for A are: "mol/liter/s" or "mol/cm3/s"
// Units allowed for E are: "kcal/mol", "cal/mol", "kJ/mol", or "J/mol"

Unit:
A: mol/cm3/s
E: kcal/mol

Reactions:
// The format is:
// R1 + R2 + R3 <=> P1 + P2 + P3        A       n       E       dA      dn      dE
//      where R1, R2, R3, P1, P2, P3 are species; A, n, and E are the Arrhenius
//      parameters, and dA, dn, dE are the errors in those parameter (normally
//      additive, but can also be multiplicative if a * comes before the number).
//      A "<=>" or "=" represents a reversible reaction and a
//      "=>" or "->" represents an irreversible reaction.
O2 + CO = CO2 + O 1.26E13 0.00 196.90 *1.7 0 0
```

3. The `pdepreactions.txt` file defines pressure-dependent reactions. The type of pressure-dependence RMG currently handles is: reactions with a third-body (bath gas), Lindemann expressions, and Troe expresions. A sample file, is listed below.

   The first line defines the reaction using the same format as in `reactions.txt`. The notable exception is the presence of the (+m) or +M in the reaction line.

   The next (optional) line lists collision efficiencies for various bath-gas species that scale the concentrations of particular species when calculating the total bath gas concentration. In the example below, CH4 is particularly effective as a 3rd body, and its effective concentration is tripled. If a species is not included in the list, the default collision efficiency is one.

   ```
   // Define the units
   // Units allowed for A are: "mol/liter/s" or "mol/cm3/s"
   // Units allowed for E are: "kcal/mol", "cal/mol", "kJ/mol", or "J/mol"

   Unit:
   A: mol/cm3/s
   E: kJ/mol

   Reactions:
   CO + O + M = CO2 + M 1.54E15 0.00 12.56 *1.2 0 0
   N2/0.4/ O2/0.4/ CO/0.75/ CO2/1.5/ H2O/6.5/ CH4/3.0/ C2H6/3.0/ AR/0.35/
   //   the first line defines the reaction and Arrhenius parameters,
   //   while the second gives the scaling factors for different bath gas species
   //   which contribute to [M].
   ```

   The next (optional) line specifies the low-pressure limit Arrhenius parameters, in the order A, n, Ea, as used in a Lindemann-type expression.

   ```
   O + CO (+M) <=> CO2 (+M)     1.800E+10     .000    2385.00   0.0     0.0     0.0
   H2/2.00/ O2/6.00/ H2O/6.00/ CH4/2.00/ CO/1.50/ CO2/3.50/ C2H6/3.00/ Ar/ .50/
   LOW/ 6.020E+14     .000    3000.00/
   //   the first two lines are similar to the third-body reaction format
   ```

```
//   the next line specifies the low pressure limit Arrhenius parameters
```

The final (optional) line specifies the Troe parameters, in the order of a, T***, T*, and T**. Note: The T** star parameter is optional.

```
C2H2 + H (+M) = C2H3 (+M) 8.43E12 0.00 10.81 *1.2 0 0
N2/0.4/ O2/0.4/ CO/0.75/ CO2/1.5/ H2O/6.5/ CH4/3.0/ C2H6/3.0/ AR/0.35/
LOW / 3.43E18 0.0 6.15 /
TROE / 1 1 1 1231 /
//   the first three lines are similar to the Lindemann reaction format
//   the next line specifies the 3 or 4 Troe parameters
//   in the order: a, T***, T*, T** (the last parameter is optional).
```

CHAPTER

TWELVE

REPRESENTING OXYGEN

Special care should be taken when constructing a mechanism that involves molecular oxygen. The ground electronic state of molecular oxygen, $^3\Sigma_g^-$, does *not* contain a double bond, but instead a single bond and two lone electrons. In RMG's adjaceny list notation the ground state of oxygen is represented as

```
(1) O2 0.1 (mol/cm3)
1 O 1 {2,S}
2 O 1 {1,S}
```

You should use the above adjacency list to represent molecular oxygen in your condition files, seed mechanisms, etc. The triplet form is 22 kcal/mol more stable than the first singlet excited state, $^1\Delta_g$, which does contain a double bond. The adjacency list for singlet oxygen is

```
(1) O2 0.1 (mol/cm3)
1 O 0 {2,D}
2 O 0 {1,D}
```

Selecting the correct structure for oxygen is important, as the reactions generated from a double bond are significantly different than those generated from a radical or diradical. For example, the reaction

$$CH_4 + O_2 \rightarrow CH_3 + HO_2$$

would occur for both triplet and singlet oxygen, but in entirely different families. For triplet oxygen the above represents a hydrogen abstraction, while for singlet oxygen it represents the reverse of a disproportionation reaction.

Previous versions of RMG databases were ambiguous as to the treatment of oxygen, with both ChemGraphs above representing ground-state oxygen. The database provided with RMG 3.2 has been modified to make all of the oxygen-related chemistry that was present in RMG databases consistent with the single-bonded biradical representation.

Conversion between triplet and singlet forms is possible through the primary reaction library `OxygenSingTrip`; the reactions involved are very slow, however, and are likely to be absent from any mechanisms generated. At this point, no other reactions of singlet oxygen have been included in RMG.

CHAPTER

THIRTEEN

ESTIMATING SPECIES THERMOCHEMISTRY

Species thermochemistry can be estimated based on 2 approaches: 1. Benson group additivity (GA) 2. On-the-fly quantum-chemical methods based on the explicit 3D structure of the molecule.

For the on-the-fly methods, a number of QM packages [G03, OpenMopac, MM4] and a number of levels of theory [PM3, MM4, MM4HR] are supported. Since the accuracy of Benson GA for acyclic molecules is generally superior than for the QM methods, acyclic molecules are always estimated through GA.

For cyclic molecules 3 different strategies can be chosen: 1. BensonOnly 2. QMforCyclics 3. Hybrid

The BensonOnly strategy only uses the Benson Group Additivity framework to estimate thermochemistry of species. For compounds containing one or more rings, it identifies the smallest set of smallest rings in the molecules and perceives whether fused ring atoms are present, that indicate that a polycyclic ring system is present. Then, the databases of ring corrections and polycyclic ring corrections are used to search an adequate (polycyclic) ring correction. If an apt ring correction is not found, a warning is printed, but the thermochemistry without the ring correction is kept.

The QMforCyclics strategy preferentially choses the on-the-fly semi-empirical methods to estimate thermochemistry. If repeated failures occur for the QM methods, this strategy falls back to the BensonOnly strategy.

In the hybrid strategy, first, the Benson databases are searched for an adequate (polycyclic) ring correction. If such a correction is not found, this strategy falls back to the QMforCyclics strategy.

The thermochemistry calculated from on-the-fly quantum methods are now saved to $RMG_QM_LIBRARY, which defaults as QMThermoLibrary in the working directory. These results can be inputted into subsequent RMG runs by copying QMThermoLibrary from a previous run into the working directory of the new run.

The `ThermoDataEstimator` utility produces thermochemistry estimates without running RMG.

13.1 Instructions for Use

1. Create a text file named `input.txt` in any directory that you like. The text file should begin with a block to specify the desired Database.

The next section allows to use on-the-fly generation QM techniques to estimate TD properties, rather than group additivity methods.

The first flag defines the strategy to estimate thermochemistry: e.g.:

```
//thermo strategy? [BensonOnly/QMforCyclics/Hybrid]
Hybrid
```

The following lines are not required if the `BensonOnly` flag is used:

The QM package to be used, is specified: either choose "both", "gaussian03", "mopac", "mm4", or "mm4hr" e.g.:

```
//QM method: [both/gaussian03/mopac/mm4/mm4hr]
gaussian03
```

Limit the use of on-the-fly methods to only species with less than the specified number of unpaired electrons. E.g. "0", implies that radical species will not computed using on-the-fly QM methods. e.g.:

```
//maxradnumforQM?
0
```

A check of the connectivity of the passed-in 3D structure can be performed ensuring that the quantumchemical data returned by the QM package is based on a 3D structure with correct connectivity. e.g.:

```
//CheckConnectivity? [off/check/confirm]
check
```

Specify whether to keep all output files from Gaussian03 or Mopac. If no enviornment variable for qmCalculationsDir is set, the output files will be kept in $WorkDir/QMFiles.

//KeepQMFiles? [yes/no] yes

The next section allows the user to set atom constraints on the species to be processed by the ThermoDataEstimator, beyond the default values hard-coded in TDE. Only species that satisfy the set atom constraints, will be processed by TDE. Uncomment (//) the desired atom constraint, to set a particular constraint.

E.g.:

```
MaxCarbonNumberPerSpecies:     40
//MaxOxygenNumberPerSpecies:     10
//MaxRadicalNumberPerSpecies:    10
//MaxSulfurNumberPerSpecies:     10
//MaxSiliconNumberPerSpecies:    10
//MaxHeavyAtomNumberPerSpecies: 100
//MaxCycleNumberPerSpecies:      10
END
```

The next field should be the PrimaryThermoLibrary, which may be left empty if desired:

```
Database: RMG_database
```

```
PrimaryThermoLibrary:
END
```

Next, in the text file, create the adjacency list(s) corresponding to the molecule(s) of interest using the same syntax used to define a species in the `condition.txt` input file (see RMG manual). Hydrogens can be omitted for simplicity. For example, the adjacency list for 2,2,4,4-tetramethylpentane could be written as:

```
1 C 0 {2,S}
2 C 0 {1,S} {3,S} {4,S} {5,S}
3 C 0 {2,S}
4 C 0 {2,S}
5 C 0 {2,S} {6,S}
6 C 0 {5,S} {7,S} {8,S} {9,S}
7 C 0 {6,S}
8 C 0 {6,S}
9 C 0 {6,S}
```

An complete example can be found in the `examples/ThermoDataEstimator` directory.

2. At the command prompt, change directory to the location of the `input.txt` file and then, if you are using a unix-based operating system, execute the command

```
$ java -classpath $RMG/bin/RMG.jar ThermoDataEstimator input.txt > RMG.log
```

Or if you are using a Windows operating system you can instead run the batch file `ThermoDataEstimator.bat`. Either double-click it or at the command propmpt type:

```
> ThermoDataEstimator.bat
```

The output of the program will automatically be written to `RMG.log`

3. The program will read RMG's thermodynamics database, count groups, and output the NASA polynomials in CHEMKIN format, as well as another set of values labeled ThermoData. The format for ThermoData is

$$\Delta H_{f,298}\ S_{298}\ C_{p,300}\ C_{p,400}\ C_{p,500}\ C_{p,600}\ C_{p,800}\ C_{p,1000}\ C_{p,1500}$$

Units are kcal/mol for $\Delta H_{f,298}$ and cal/mol*K for the other quantities (entropy and heat capacity). The symmetry number (including contributions from internal rotations) will also be displayed.

CHAPTER

FOURTEEN

ESTIMATING SPECIES TRANSPORT PROPERTIES

The `TransportDataEstimator` utility produces group additivity-based transport property estimates - in particular, the Lennard-Jones sigma and epsilon parameters - without running RMG. As with the transport estimates used by RMG, the `TransportDataEstimator` utility bases its estimates on group values located in the RMG database.

14.1 Instructions for Use

1. Create a text file named `input.txt` in any directory that you like. The text file should begin with a block to specify the desired Database. The next field should be the PrimaryTransportLibrary, which may be left empty if desired:

   ```
   Database: RMG_database

   PrimaryTransportLibrary:
   END
   ```

   Next, in the text file, create the adjacency list(s) corresponding to the molecule(s) of interest using the same syntax used to define a species in the `condition.txt` input file (see RMG manual). Hydrogens can be omitted for simplicity. For example, the adjacency list for 2,2,4,4-tetramethylpentane could be written as:

   ```
   Tetramethylpentane_2244
   1 C 0 {2,S}
   2 C 0 {1,S} {3,S} {4,S} {5,S}
   3 C 0 {2,S}
   4 C 0 {2,S}
   5 C 0 {2,S} {6,S}
   6 C 0 {5,S} {7,S} {8,S} {9,S}
   7 C 0 {6,S}
   8 C 0 {6,S}
   9 C 0 {6,S}
   ```

2. At the command prompt, change directory to the location of the `input.txt` file and then, if you are using a unix-based operating system, execute the command

```
$ java -classpath $RMG/bin/RMG.jar TransportDataEstimator input.txt > RMG.log
```

If you are using a Windows operating system you can instead run the batch file `TransportDataEstimator.bat`. Either double-click it or at the command propmpt type:

```
> TransportDataEstimator.bat
```

The program's output is automatically written to `tran.dat`

3. The program will read RMG's transport database, count groups, and output the species estimated transport properties in CHEMKIN format.

The first entry is the species name (as supplied by you, assuming the number of characters is less than 16). The next entry is the RMG estimate on whether the species is an atom (0), linear (1), or nonlinear (2). The next two entries are the RMG estimates for the Lennard-Jones epsilon and sigma parameters; the units of epsilon are Kelvin while the units of sigma are Angstroms. The next three entries - the dipole moment, polarizability, and rotational relaxation collision number at 298 K - are default values and are not estimated by RMG; these entries are included so that the output `tran.dat` is CHEMKIN-readable. All remaining information is comments regarding how RMG estimated the Lennard-Jones properties, including the estimated critical properties and which groups in the `RMG_databases/transport_groups` were matched.

CHAPTER

FIFTEEN

IDENTIFYING ALL REACTIONS FOR A SET OF SPECIES

The `PopulateReactions` module accepts a list of species as input (in the form of adjacency lists) and outputs a list of all possible reactions between those species, according to the RMG reaction family templates (located at `%rmg%\databases\RMG_database\kinetics_groups`) and any user-specified reaction library templates.

Different usages for this module:

1) Determine if the RMG reaction family templates can form your reaction of interest. If not, this reaction would be an ideal candidate to place in a *Primary Reaction Library* (page 46).

2) Determine if RMG is properly parsing your *Primary Kinetic Library* (page 45) and/or *Primary Reaction Library* (page 46) without having to run an entire RMG simulation.

3) Obtain more detailed comments for RMG-generated kinetics. Suppose you ran a RMG simulation with the *Verbose field* (page 48) set to off and are now interested in how RMG estimated a certain reaction's kinetics. Instead of re-running the entire RMG simulation, one could place the species of interest in this module, with the Verbose field turned on, and obtain the detailed kinetics quickly.

4) Obtain the pressure-dependent kinetics for an entire pressure-dependent network. RMG will output the pressure-dependent kinetics for the "core" reactions only; if one wished to have the entire network's kinetics at hand, this module will run the single pressure-dependent network, without having to run the entire RMG job. Furthermore, the user can test how sensitive a pressure-dependent network's kinetics are to the different *collisional approximations* (page 39) and/or to the *number of grains used* (page 41) to solve the network.

In essence, the PopulateReactions module runs a single RMG "enlarging" step, without solving any ordinary differential equations, nor distinguishing species and reactions as "core" or "edge".

15.1 Instructions for Use

1. Create a text file in any directory that you like (for the purposes of this example, we'll create a file named `input.txt`). A sample file is shown below:

```
Database: RMG_database

PrimaryThermoLibrary:
```

```
Name: RMG_Default
Location: primaryThermoLibrary
END

PrimaryTransportLibrary:
Name: GRI-Mech-3
Location: GRI-Mech3.0
END

TemperatureModel: Constant (K) 1200
PressureModel: Constant (atm) 1

SpeciesStatus:

nButanol (molecule/cm3) 0.01
1 C 0 {2,S}
2 C 0 {1,S} {3,S}
3 C 0 {2,S} {4,S}
4 C 0 {3,S} {5,S}
5 O 0 {4,S}

CH3 (molecule/cm3) 0.01
1 C 1

END

BathGas:
N2 (molecule/cm3) 0.99
END

SpectroscopicDataEstimator: FrequencyGroups
PressureDependence: ReservoirState
PDepKineticsModel: Chebyshev
//DecreaseGrainSize: yes

PrimaryKineticLibrary:
//Name: PrIMe-recommended
//Location: GRI-Mech3.0
END

ReactionLibrary:
END

//Verbose: off
```

The formatting of the `input.txt` file is very similar to the formatting of the `condition.txt` file for an RMG simulation. For more details on the meaning of each field, and possible values to supply, see the instructions for *Creating a Condition File Manually* (page 31).

If the PressureDependence field is set to "off," this module will try to match each species (e.g. A, B, C, etc.) against every unimolecular RMG reaction family template (and every user-specified Reaction Library template) and each combination of species (e.g. A+A, A+B, B+A, etc.) against every bimolecular RMG reaction family template (and user-specified Reaction Library template). All reactions created by RMG will

be reported in the output file, with modified Arrhenius kinetics.

If the PressureDependence field is not set to "off," this module behaves in the same manner as described above, i.e. finding all possible reactions between all combinations of species, and then sends every potential pressure-dependent reaction (i.e. any reaction that has either only one reactant or one product) to the Fame executable. The pressure-independent reactions (e.g. H-Abstraction and Disproportionation reactions) will still be reported with modified Arrhenius parameters in the output files, but the pressure-dependent reactions will have formatted pressure-dependent kinetics (based on the PDepKineticsModel field).

Note: The pressure-dependence feature assumes all unimolecular isomers have been supplied to the input file. For example, if one wished to have the "entire" pressure-dependent network for the .OOCH(CH3)2 potential energy surface, the following species would need to be supplied in the input file:

```
OOCH(CH3)2 (mol/cm3) 1
1 O 1 {2,S}
2 O 0 {1,S} {3,S}
3 C 0 {2,S} {4,S} {5,S}
4 C 0 {3,S}
5 C 0 {3,S}

HOOCH(CH2)(CH3) (mol/cm3) 1
1 O 0 {2,S}
2 O 0 {1,S} {3,S}
3 C 0 {2,S} {4,S} {5,S}
4 C 0 {3,S}
5 C 1 {3,S}
```

With the first species, this module will find the following reactions (among others):

```
O2 + CH3-.CH-CH3 --> OOCH(CH3)2
OOCH(CH3)2 --> HOOCH(CH2)(CH3)
```

The second species yields the following reactions (among others):

```
HOOCH(CH2)(CH3) --> OOCH(CH3)2  (this is a duplicate, and the module will recognize this)
HOOCH(CH2)(CH3) --> HO2 + C3H6
```

Thus, the "wells" sent to Fame (among others) will be:

```
O2 + CH3-.CH-CH3
OOCH(CH3)2
HOOCH(CH2)(CH3)
HO2 + C3H6
```

Fame will then return (and this module will report in the output file) 6 sets of kinetics:

```
A(+m)=B(+m)
A(+m)=C(+m)
A(+m)=D(+m)
B(+m)=C(+m)
B(+m)=D(+m)
C(+m)=D(+m)
```

(Consider O2 + CH3-.CH-CH3 to be "A", OOCH(CH3)2 to be "B", etc.). Thus, the "entire" network and its kinetics will be present in the output file.

However, if only the OOCH(CH3)2 species is supplied, the "wells" sent to Fame will be:

```
O2 + CH3-.CH-CH3
OOCH(CH3)2
HOOCH(CH2)(CH3)
```

Recall that this module (essentially) does one RMG enlarging step; reacting from HOOCH(CH2)(CH3) to C3H6 + HO2 will involve a second enlarging step. So, if you believe this module is not returning the entire network of reactions, please consider if you have supplied all unimolecular "wells" in the input file.

---

2. At the command prompt, change to the directory containing the `input.txt` file and then execute the command

```
java -classpath "%rmg%"\bin\RMG.jar PopulateReactions input.txt
```

For linux users:

```
java -classpath $RMG/bin/RMG.jar PopulateReactions input.txt
```

To have the output written to file instead of to the screen, use the syntax

```
java -classpath "%rmg%"\bin\RMG.jar PopulateReactions input.txt > output.txt
```

where `output.txt` is the name of the file to be written.

3. Every time a species is found in the input file, the module will:
    - Read in the name and adjacency list of the species
    - Attempt to react the just-read-in species against each of the reaction families located in the `%rmg%\databases\RMG_database\kinetics_groups` directory AND against each of the reactions in all user-specified Reaction Libraries
    - Store the results

The reaction families the module will attempt to react the just-read-in species against are either unimolecular or bimolecular.

- If the reaction family is unimolecular, the module attempts to react the current species
- If the reaction family is bimolecular (e.g. A+B–>products), the module assigns the just-read-in species as A and iterates over all previously read-in species (including the just-read-in species) when assigning B. The module then assigns the just-read-in species as B and iterates over all previous species for A.

All reactions generated are reported in the `PopRxnsOutput_Rxns.txt` file. The top of the file displays the units for the modified Arrhenius parameters reported

Each high-pressure limit line after this has the following format (in general):

B+C–>D+F A n Ea Rxn_family exact/estimate: Node1 Node2

where:

- A, n, and Ea are the modified Arrhenius parameters

- Rxn_family is the reaction family template used to generate this reaction. The "Rxn_family" is a folder in the `%rmg%\databases\RMG_database\kinetics` directory
- Node1/Node2 are the reactive subgroups of the species that RMG identified
- exact/estimate states whether the Rxn_family folder's `rateLibrary.txt` file contained data for the Node1-Node2 combination. If so, the kinetics reported are found, exactly as written, in the file. If not, an averaging scheme was performed to arrive at the reported numbers.

The identities of B, C, D, and F are located in the `PopRxnsOutput_Spcs.txt` file, in the form of adjacency lists.

Note: These high-pressure limit reactions appear at the bottom of the `PopRxnsOutput_Rxns.txt` file when the PressureDependence field is not set to "off".

Each pressure-dependent network has the following format:

```
!PDepNetwork
!       deltaEdown = 5.514552177052681 kJ/mol
!       bathgas MW = 28.01 amu
!       bathgas LJ sigma = 3.681E-10 meters
!       bathgas LJ epsilon = 1.263615E-21 Joules
!Here are the species and their thermochemistry:
!       nButanol(1)             -66.06  85.94   26.48   33.09   39.02   44.15   52.29   58
!       CH3(2)                  34.81   46.37   9.14    10.18   10.81   11.34   12.57   13
!       C3H7OJ(3)               -12.13  79.93   20.21   24.78   28.86   32.4    38.06   42
!       C4H9OJ(4)               -17.06  89.35   25.71   31.73   37.11   41.75   49.13   54
!       HJ(5)                   52.1    27.42   4.97    4.97    4.97    4.97    4.97    4.9
!       C2H5J(6)                28.6    59.87   11.73   14.47   17.05   19.34   23.02   25
!       C2H5OJ(7)               -7.2    70.51   14.71   17.83   20.61   23.05   26.99   29
!       C4H9OJ(8)               -19.71  91.07   24.94   30.32   35.53   40.25   47.94   53
!       C3H7J(9)                23.67   69.29   17.11   21.27   25.14   28.53   33.95   38
!       CH3OJ(10)               -3.59   58.19   10.74   12.52   13.96   15.19   17.1    18
!       C4H9OJ(11)              -19.71  90.38   24.98   30.76   35.92   40.76   48.54   53
!       C4H9J(12)               18.74   78.71   22.61   28.22   33.39   37.88   45.02   50
!       HOJ(13)                 9.4     43.91   7.14    7.07    7.05    7.06    7.15    7.3
!       C4H9OJ(14)              -22.77  86.86   27.13   33.08   38.27   42.72   49.77   55
!       C4H9OJ(15)              -14.1   84.48   25.5    31.79   37.41   42.26   49.91   55
!       C2H6O(16)               -56.2   67.1    15.48   19.19   22.52   25.45   30.15   33
!       C2H4(17)                12.52   52.47   10.2    12.72   15.02   17.0    20.14   22
!       C4H8(18)                -0.11   73.61   20.57   26.09   31.04   35.28   41.96   46
!       H2O(19)                 -57.8   45.13   8.03    8.19    8.42    8.68    9.26    9.8
!Here are the path reactions and their high-P limit kinetics:
!       CH3(2)+C3H7OJ(3)=nButanol(1)            3.370e+13       0.00    0.00
!       C4H9OJ(4)+HJ(5)=nButanol(1)             1.000e+14       0.00    0.00
!       C2H5J(6)+C2H5OJ(7)=nButanol(1)          1.150e+13       0.00    0.00
!       C4H9OJ(8)+HJ(5)=nButanol(1)             2.000e+13       0.00    0.00
!       C3H7J(9)+CH3OJ(10)=nButanol(1)          1.000e+13       0.00    0.00
!       C4H9OJ(11)+HJ(5)=nButanol(1)            2.000e+13       0.00    0.00
!       C4H9J(12)+HOJ(13)=nButanol(1)           7.700e+13       0.00    0.00
!       C4H9OJ(14)+HJ(5)=nButanol(1)            5.000e+13       0.00    0.00
!       C4H9OJ(15)+HJ(5)=nButanol(1)            1.000e+13       0.00    0.00
!       C2H6O(16)+C2H4(17)=nButanol(1)          1.630e+02       2.92    44628.57
```

```
!       C4H8(18)+H2O(19)=nButanol(1)              2.270e+02        2.74     56900.00
```

The first four lines after "PDepNetwork" describe properties of the Bath Gas that was utilized in Fame. In general, the Lennard-Jones (LJ) sigma and epsilon parameters are estimated by RMG. However, a user may use their preferred value for any species by supplying a *Primary Transport Library* (page 33) to the input file.

The next set of information is the name and thermochemistry of every species present in the pressure-dependent network; the thermochemistry is reported as:

```
Hf(298 K)        S(298 K)        Cp(300, 400, 500, 600, 800, 1000, 1500 K)        Source_of_
```

The units of Hf are kcal/mol and the units for S and Cp are cal/mol-K.

The last section contains the reactions sent to the Fame executable, and their high-pressure limit kinetics.

Following this information section is every pressure-dependent reaction in the network, along with their respective pressure-dependent kinetics. The formatting of this set of output is identical to that found in the RMG-generated `chem.inp` file.

CHAPTER

SIXTEEN

CONVERTING BETWEEN DIFFERENT SPECIES IDS

Instead of constructing an adjacency list by hand, new modules have been built to convert IUPAC **In**ternational **Ch**emical **I**dentifiers (InChIs) and .mol files to adjacency lists.

Note: The InChI executable must be present in the `%rmg%\bin` directory for the InChI2AdjList and AdjList2InChI modules to function.

16.1 InChI2AdjList

To convert an InChI (or a series of InChIs) into an adjacency list, create a text file in any directory you like (for the purposes of this example, we'll create a file named `listOfInChIs.txt`). List each InChI in this new file, separated by a carriage return. If you'd like to identify the InChI with a particular name (e.g. the InChI for 1,3-hexadiene as HXD13), place this name before the InChI; if a line only contains the InChI, the default name of this species will be the InChI itself.

An example of the `listOfInChIs.txt` file is:

```
HXD13    InChI=1/C6H10/c1-3-5-6-4-2/h3,5-6H,1,4H2,2H3
CH4      InChI=1/CH4/h1H4
H2       InChI=1/H2/h1H
```

Running the following command will generate a file (`adjList_output.txt`) which contains a list of species, in the form of adjacency lists:

```
java -classpath "%rmg%"\bin\RMG.jar InChI2AdjList listOfInChIs.txt
```

16.2 Mol2AdjList

To convert a series of .mol files into a list of adjacency lists, place the list of .mol files in any directory you like (for the purpose of this example, we'll create a file named `temp.mol` in a folder named `directoryOfMolFiles`). An example of a .mol file (for 1,3-hexadiene) is:

```
Structure #1
  InChI v1 SDfile Output

  6  5  0  0  0  0  0  0  0  0  1 V2000
    0.0000    0.0000    0.0000 C   0  0  0     0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0     0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0     0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0     0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0     0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0     0  0  0  0  0  0
  1  3  2  0  0  0  0
  2  4  1  0  0  0  0
  3  5  1  0  0  0  0
  4  6  1  0  0  0  0
  5  6  2  0  0  0  0
M  END
$$$$
```

Passing the location of this directory to the Mol2AdjList console, using the following command, generates the adjacency list in a file named `adjList.txt`:

```
java -classpath "%rmg%"\bin\RMG.jar Mol2AdjList directoryOfMolFiles > adjList.txt
```

16.3 AdjList2InChI

To convert an adjacency list (or series of adjacency lists) to an InChI, create a text file in any directory you like (for the purposes of this example, we'll create a file named `listOfAdjLists.txt`). List each adjacency list in this new file, separated by a carriage return. Furthermore, you must identify each adjacency list with a name (e.g. HXD13 for 1,3-hexadiene); place this name before the adjacency list.

An example of the `listOfAdjLists.txt` file is:

```
HXD13
1 C 0 {2,D}
2 C 0 {1,D} {3,S}
3 C 0 {2,S} {4,D}
4 C 0 {3,D} {5,S}
5 C 0 {4,S} {6,S}
6 C 0 {5,S}

CH4
1 C 0

H2
1 H 0 {2,S}
2 H 0 {1,S}
```

Running the following command will generate a file (`inchi_output.txt`) which contains a list of InChIs, identified by the name given in the input file:

```
java -classpath "%rmg%"\bin\RMG.jar AdjList2InChI listOfAdjLists.txt
```

CHAPTER

SEVENTEEN

THEORY AND IMPLEMENTATION DETAILS

A schematic overview of RMG is presented below.

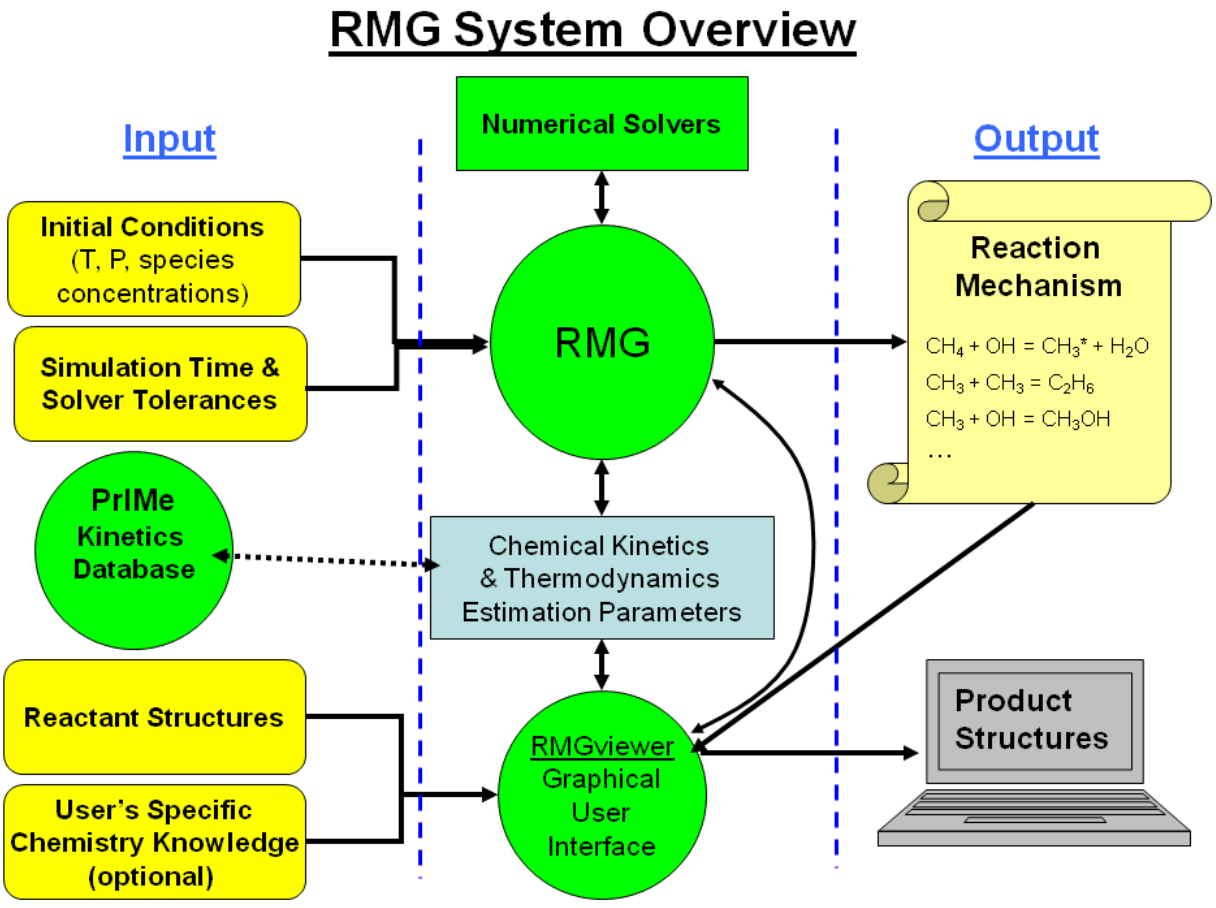


Figure 17.1: A schematic overview of RMG

Many of the details of RMG's original implementation (Version 1) are described in [JingThesis] (page 156). For a more recent discussion of RMG, see [GreenChapter] (page 156).

Details of certain aspects of RMG are discussed in greater detail below.

17.1 Rate-Based Model Enlarger

To construct a mechanism, the user must specify an initial set of species and the initial conditions (temperature, pressure, species concentrations, etc.). RMG reacts the initial species in all possible ways according to its known reaction families, and it integrates the model in time. RMG tracks the rate (flux) at which each new "edge" species is produced, and species (and the reactions producing them) that are produced with significant fluxes are incorporated into the model (the "core"). These new core species are reacted with all other core species in the model, to generate a new set of edge species and reactions. The time-integration restarts, and the expanded list of edge species is monitored for significant species to be included in the core. The process continues until all significant species and reactions have been included in the model. The user is free to vary the definition of a "significant" rate to refine the mechanism as desired. For a more detailed description on rate-based model enlargement, please see [Susnow1997] (page 155).

17.2 Thermodynamics Estimation Using Group Additivity

Benson's Group Additivity approach, used by RMG, divides a molecule into functional groups, and the contribution of each functional group to the overall thermodynamics is included. RMG's kinetic rules use a similar strategy: the reaction rate coefficient depends on the two (or more) functional groups involved in the reaction. The rate coefficient rule database uses an innovative tree structure that allows users to include ever-more specific defintions of the functional groups and the kinetics involving them. In both cases, users can easily modify and expand these databases to suit their needs. In principle, the trees could be expanded until all possible molecules and data for the reactions between all possible pairs of molecules have been included (of course, in practice, the database is much more limited). The kinetics database includes 19 reaction types ("families"), including hydrogen abstractions, cycloadditions, disproportionation reactions, radical recombination reactions, intramolecular rearrangements, etc.

Each reaction family includes data on many possible functional group pairings involved in the reaction, for a total of roughly 1000 rate rules. When a perfect match for a given reaction is not found in the database, RMG estimates the rate coefficient parameters based on "nearby" rate coefficient rules that are similar to the desired one. In addition to modifying the RMG databases, the user may also enter thermodynamic or kinetic information for specific molecules/reactions into Primary Thermo/Kinetic Libraries which supersede the estimated values. In this manner, precise data for well-studied reactions/species can be included.

17.3 Pressure Dependence in RMG

One of the new features in RMG is the ability to calculate pressure-dependent rate constants. This appendix provides a brief background on the subject, followed by an explanation of how RMG calculates these rate constants. For a more detailed description on pressure-dependent rate constants, please consult the excellent texts on unimolecular reaction theory by Gilbert and Smith [Gilbert1990] (page 155); Holbrook, Pilling, & Robertson [Holbrook1996] (page 155) ; and Forst [Forst2003] (page 155).

Consider a molecule in a vibrationally excited state. Under non-pressure dependent conditions, the excess vibrational energy is removed via non-elastic collisions with other molecules, and the rate at which this

collisional relaxation occurs is significantly faster than any competing chemical reactions. Under certain circumstances, however, this is not the case: The vibrationally excited molecule can either isomerize or dissociate on a time scale competitive with or even faster than the time scale of collisions. When that occurs, the phenomenological rate constant from the initial molecule to the possible product channels will depend on both temperature and pressure.

Two conditions can cause pressure dependence: low pressures and high temperatures. The first condition is the simplest, and historically most discussions on the subject of pressure dependence focus on unimolecular reactions at low pressures. The collision frequency is directly proportional to the pressure, so as the pressure is decreased, the rate of collisional energy transfer decreases. Eventually the pressure becomes low enough that the rate of chemical reaction becomes faster than the collision rate.

Chemical activation is not limited to these low-pressure cases, however. Indeed, under the high temperatures of combustion-relevant conditions, many bimolecular reactions at atmospheric pressure will have multiple product channels with pressure-dependent branching ratios. When two molecules collide, the initial adduct contains excess energy that is rapidly redistributed among the vibrational modes. As the temperature is increased, a greater percentage of the adduct population is located at higher vibrational energy levels. As more of the population shifts to higher energy levels, a greater percentage of the population can undergo the isomerization and/or dissociation reactions. If the temperature is high enough, most of the population will be lost to chemical reaction before it can be collisionally stabilized.

The difficulty, of course, is knowing (i) when a reaction becomes pressure dependent, and (ii) how to quantify the pressure dependence if it occurs. To answer these questions, we need to solve the master equation.

17.3.1 Formulation of the Master Equation

Begin with the number density of species i with energy between E and $E + dE$, $n_i(E)dE$. In a chemically activated or pressure-dependent system, the energy-dependent population can undergo the following possibilities: it can gain or lose energy by colliding with other molecules, it can be formed by the association reaction of the reactants, it can decompose back to the reactants, it can isomerize to form other well species, or it can decompose to form bimolecular products. Thus,

$$
\begin{aligned}
\frac{dn_i\left(E\right)}{dt} &= \text{Gain to } n_i\left(E\right) \text{ via collisional energy transfer from } n_i\left(E'\right) \\
&- \text{loss from } n_i\left(E\right) \text{ via collisional energy transfer to all other } n_i\left(E'\right) \\
&+ \text{gain to } n_i\left(E\right) \text{ via association of reactants } n_R, n_m \\
&- \text{loss from } n_i\left(E\right) \text{ via decomposition back to reactants } n_R, n_m \\
&+ \text{gain to } n_i\left(E\right) \text{ via isomerization from } n_j\left(E\right) \\
&- \text{loss from } n_i\left(E\right) \text{ via isomerization to } n_j\left(E\right) \\
&- \text{loss from } n_i\left(E\right) \text{ via decomposition to bimolecular products } p_i
\end{aligned}
$$

Mathematically, we have

$$\begin{aligned}\frac{dn_i(E)}{dt} &= \omega\int_0^\infty P_i\left(E,E'\right)n_i\left(E'\right)dE' - \omega n_i(E)\\ &+ n_R n_m K_{eq}k_{dis}(E)F_i(E)\delta_{Rmi} - k_{dis}(E)n_i(E)\delta_{Rmi}\\ &+ \sum_{j\neq i}^{N_{wells}} k_{ij}(E)n_j(E) - \sum_{j\neq i}^{N_{wells}} k_{ji}(E)n_i(E)\\ &- \sum_{p_j}^{N_{prod}} k_{p_j i}(E)n_i(E)\end{aligned} \tag{17.1}$$

The corresponding equations for the reactants are:

$$\begin{aligned}\frac{dn_R}{dt} = \frac{dn_m}{dt} &= \sum_{i=1}^{N_{wells}}\delta_{Rmi}\int_{E_{i0}}^\infty k_{dis}(E)n_i(E)dE\\ &- \sum_{i=1}^{N_{wells}}\delta_{Rmi}\int_{E_{i0}}^\infty K_{eq}k_{dis}(E)F_i(E)n_R n_m dE\end{aligned} \tag{17.2}$$

The units of the number density $n_i(E)\,dE$ are [particle/volume]. ω is the collision frequency [s^{-1}], $P_i(E,E')\,dE$ is the probability that a particle with energy E' will have energy E after collision [unitless], $k_{ij}(E)$ is the microcanonical rate constant for the isomerization from well j to well i [s^{-1}], K_{eq} is the equilibrium constant between the reactants R and m and the i^{th} well [volume/particle], $k_{dis}(E)$ is the microcanonical rate constant for the dissociation back to the reactants [s^{-1}], $\rho_i(E)$ is the density of states [$energy^{-1}$], $Q_i(T)$ is the partition function for the active degrees of freedom [unitless], n_R and n_m are the number density of the reactants [particle/volume], and $k_{p_j i}(E)$ is the microcanonical rate constant for the dissociation from well i to the j^{th} product channel [s^{-1}]. The delta function δ_{Rmi} is there to ensure that only wells with direct access to the reactants can be formed from or dissociate back to the reactants. In principle, the energy-dependent populations are a series of delta peaks and not a continuous function; in practice, however, at energy levels relevant for our calculations, the spacing between delta functions is so small that the populations can be approximated by a continuous function.

17.3.2 Simplifying Assumptions and Solution

At this point we have $N_{wells}+2$ coupled, non-linear integro-differential equations. We would prefer a system of linear ODEs. To simply the system of equations, we start with two assumptions:

1. Energy is discretized
2. n_m is constant

The first assumption makes the system of equations numerically tractable; by discretizing the energy, we replace the integrals with summations, which yields $N_{\text{energy levels}} \times N_{\text{wells}} + 2$ coupled, non-linear ordinary differential equations. The second assumption assumes that one of the reactants is in great excess over the other (e.g. a stable species rather than a radical). This assumption linearizes the system of equations, yielding $N_{\text{energy levels}} \times N_{\text{wells}} + 1$ coupled linear ODEs. Two methods are commonly employed to solve this system of ODEs: eigenvalue decomposition, and a stochastic method. For eigenvalue decomposition, the code VariFlex [VariFlex] (page 155) is recommended; for stochastic simulations, the code MultiWell [MultiWell] (page 155) [Barker2001] (page 156) is recommended.

Next, we make two additional assumptions:

3. The d/dt is equal to zero
4. n_R is constant

The third assumption assumes that the lifetime of the vibrationally excited intermediates is significantly less than the phenomenologically relevant time scale of the reaction. This assumption converts the system of ODEs into a single matrix equation of rank $N_{\text{energy levels}} \times N_{\text{wells}} + 1$. The fourth assumption reduces the order of the rank by one, and more importantly it ensures that the matrix equation has a non-zero solution. Conceptually, it suggests that the reaction could be modeled as a perfectly stirred reactor with a constant inlet stream, rather than as a batch reactor. With these two additional assumptions, we now have a steady-state master equation.

In the current implementation we must make one last assumption:

5. The collisional energy transfer, $\omega \int_0^\infty P_i\left(E, E'\right) n_i\left(E'\right) dE'$, is replaced by a modified strong collision, $\omega\beta$

This final assumption is the most drastic. It assumes that a certain fraction, $beta$, of all collisions will quench the vibrationally excited adducts. This assumption decouples all the energy levels for a given well, thereby reducing the large matrix equation into $N_{\text{energy levels}}$ individual matrix equations of rank N_{wells}. This approach is similar to the modified-strong collision used in CHEMDIS [CHEMDIS] (page 156). With these five assumptions, the individual equations simply to:

$$\begin{aligned} S_i\left(E\right) &= \left[\beta\omega + \sum_{j\neq i}^{N_{wells}} k_{ji}\left(E\right) + \delta_{Rmi} k_{dis}\left(E\right) + \sum_{p_j}^{N_{prod}} k_{p_j i}\left(E\right)\right] n_i\left(E\right) \\ &- \sum_{j\neq i}^{N_{wells}} k_{ij}\left(E\right) n_j\left(E\right) \end{aligned} \tag{17.3}$$

where $S_i\left(E\right) \equiv K_{eq} k_{dis}\left(E\right) n_R n_m \delta_{Rmi} \frac{\rho_i(E) e^{-\beta E}}{Q_i(T)}$ is the source term, which is the product of the association rate, $K_{eq} k_{dis}\left(E\right) n_R n_m \delta_{Rmi}$, and the equilibrium energy distribution $\frac{\rho_i(E) e^{-\beta E}}{Q_i(T)}$.

The terms in square brackets on the right-hand side are the loss channels; these terms appear in the diagonal of the matrix. The other term on the right-hand side is the gain to a well from the other wells; these rates appear in the off-diagonal positions. To calculate the phenomenological rate constants, the above equation is solved for each energy level. The phenomenological rate constant for a particular channel is the dot product of the steady-state population vector and the vector containing the microcanonical rate constant for that channel.

17.3.3 Supporting Equations

Collision Frequency

The collision frequency, ω, is the product of the second-order energy-transfer rate constant and the bath gas concentration. To simply calculations, it is assumed that the energy transfer rate constant is independent of

energy and is equal to the product of the hard-sphere collision limit and the collision integral for a Lennard-Jones potential [Forst2003] (page 155):

$$\begin{aligned} \omega &= \pi d^2 \sqrt{\frac{8k_BT}{\pi\mu}} \Omega\left(T\right) \frac{P}{k_BT} \\ \Omega\left(T\right) &= \frac{1.16145}{\left(T^*\right)^{0.14874}} + \frac{0.52487}{\exp\left[0.7732T^*\right]} + \frac{2.16178}{\exp\left[2.437887\right]} \\ T^* &= \frac{k_BT}{\epsilon} \end{aligned}$$

where d is the collision diameter, μ is the reduced mass, $\Omega\left(T\right)$ is the collision integral, and ϵ is the Lennard-Jones parameter for the depth of the potential. Note that it is through the collision frequency that the pressure enters the system of equations.

Collision Efficiency

Currently RMG uses a constant collision efficiency of $\beta = 0.38$. In the future a more rigorous formula will be introduced to more accurately capture the temperature dependence of β.

Microcanonical Rate Constants

RMG currently does not have data for individual transition states. Consequently, it is not possible to calculate the microcanonical rate constants using RRKM theory. Instead, RMG uses its database of Arrhenius parameters to calculate the microcanonical rate constant using the inverse Laplace transform method. For more details on the inverse Laplace transform method, please read Forst [Forst2003] (page 155) and further references therein. One benefit of using the inverse Laplace transform method is that all the multiplicative constants will cancel. Thus, as we will see below in Section *Density of States Estimation* (page 131), we do not need to know the rotational constants and symmetry numbers for a molecule.

Given an Arrhenius expression for the high-pressure limit rate constant, the microcanonical rate constant is:

$$\begin{aligned} k\left(T\right) &= Ae^{-E_a/RT} \\ k\left(E\right) &= \begin{cases} A\dfrac{\rho\left(E-E_a\right)}{\rho\left(E\right)} & E > E_a \\ 0 & E \leq E_a \end{cases} \end{aligned} \tag{17.4}$$

where A is the Arrhenius pre-exponential factor, E_a is the Arrhenius activation energy, and R is the ideal gas constant.

For modified-Arrhenius equations, the equation above requires some modification. Specifically, the temperature dependence of the A-factor must be convoluted with the density of states. This convolution is possible only when the temperature exponent is positive. For negative temperature exponents, the inverse Laplace transform is not possible. Instead, the only option is to use an effective A-factor, $A_{eff} = AT^n$. Thus for

$n > 0$, we have:

$$\begin{aligned} k(T) &= AT^n e^{-E_a/RT} \\ \phi(E) &= \frac{1}{k_B^n \Gamma(n)} \int_0^E (E-x)^{n-1} \rho(x)\, dx \\ k(E) &= \begin{cases} A\dfrac{\phi(E-E_a)}{\rho(E)} & E > E_a \\ 0 & E \leq E_a \end{cases} \end{aligned}$$

and for $n < 0$, we have:

$$\begin{aligned} k(T) &= AT^n e^{-E_a/RT} \\ k(E) &= \begin{cases} A_{eff}\dfrac{\rho(E-E_a)}{\rho(E)} & E > E_a \\ 0 & E \leq E_a \end{cases} \end{aligned}$$

For barrierless reactions, such as radical-radical reactions and radical + O_2 reactions, the activation energy in the Arrhenius expression is often negative. For these types of reactions, variational transition state theory is required to calculate the microcanonical rate constant; unfortunately, it is impossible for RMG to implement a VTST approach. Furthermore, because there is no inverse Laplace transform for a positive exponential, the Inverse Laplace method described in the previous two sections must be altered. At best, RMG has one of two approaches. In the simplest approach RMG can calculate an effective A-factor from the product of the A-factor and the exponential term: $A_{eff} = Ae^{|E_a|/RT}$. Applying the inverse Laplace transform as before, the microcanonical rate constant will be constant with respect to energy for a standard Arrhenius equation, and it will increase slightly with energy for a modified-Arrhenius equation:

$$\begin{aligned} k(E) &= A_{eff} && \text{standard Arrhenius} \\ k(E) &= A_{eff}\frac{\phi(E)}{\rho(E)} && \text{modified Arrhenius} \end{aligned}$$

The second method makes use of the fact that the Arrhenius activation energy is relatively small for barrierless reactions (often less than 1 kcal/mol), so it performs a truncated series expansion of the exponential: $e^{|E_a|\beta} \approx 1 + |E_a|\,\beta$. This approach guarantees that the rate constant will increase with energy. However, it requires the derivative of the density of states, which must be done numerically. Applying the inverse Laplace transform to this approach yields:

$$\begin{aligned} k(E) &= A\frac{\rho(E) + |E_a|\,\frac{d\rho(E)}{dE}}{\rho(E)} && \text{standard Arrhenius} \\ k(E) &= A\frac{\phi(E) + |E_a|\,\frac{d\phi(E)}{dE}}{\rho(E)} && \text{modified Arrhenius} \end{aligned}$$

Density of States Estimation

An important quantity to the pressure dependence calculation is the density of states: the number of quantum states per unit energy. In the master equation, we are interested in the density of states for the active degrees of freedom. The active degrees of freedom are those degrees of freedom in which the energy of the excited species can be randomized; within RMG, they include the rigid-rotor harmonic-oscillator (RRHO)

vibrational frequencies, hindered internal rotors (HR), and the unique one-dimensional external rotation of the molecule (i.e. the K-rotor for a symmetric top approximation). The translational degrees of freedom and the two degenerate external rotational degrees of freedom are inactive. Furthermore, RMG assumes that each of these modes is independent. Thus, just as the partition function for two independent modes is the product of the individual partition functions, the density of states for two independent modes is the convolution integral of the respective density of states:

$$\begin{aligned} Q_{AB}(T) &= Q_A(T)\,Q_B(T) \\ \rho_{AB}(E) &= \mathcal{L}^{-1}\{Q_1 Q_2\} \\ &= \int_0^E \rho_A(E-x)\,\rho_B(x)\,dx \end{aligned} \tag{17.5}$$

To calculate the density of states, RMG assumes that each molecule may be approximated as a symmetric top, which implies that the non-degenerate rotational mode is an active degree of freedom. The density of states for this one-dimensional rotor is:

$$\rho_r(E) = \frac{1}{\sigma}\left(\frac{1}{BE}\right)^{1/2} \tag{17.6}$$

where σ is the rotational symmetry number, and B is the rotational constant. As noted above in the section on *Microcanonical Rate Constants* (page 130), these parameters do not matter, since they will cancel in the inverse Laplace transform method. For internal rotors, the density of states is more complicated, and the calculation method is described as follows. For simplicity, it is assumed that the potential for internal rotation may be approximated by a single cosine:

$$V(\theta;\sigma) = \frac{1}{2}V_0\left[1-\cos(\sigma\theta)\right]$$

where V_0 is the barrier height, θ is the dihedral angle, and σ is the number of potential minima (e.g. 3 for a methyl group). The classical partition function for a hindered rotor is:

$$\begin{aligned} Q_{\text{HR}_{\text{classical}}} &= \frac{1}{h^2}\int\int e^{-H/k_BT}dqdp \\ &= \left(\frac{\pi k_B T}{\sigma^2 B_r}\right)^2 e^{-V_0/2k_BT} I_0\left(\frac{V_0}{2k_BT}\right) \end{aligned}$$

where $B_r \equiv 8\pi^2 h^2/I_r$, h is Planck's constant, I_r is the reduced moment of inertia, and I_0 is the modified Bessel function of the first kind. The semi-classical approximation for the hindered rotor partition function is classical hindered-rotor partition function multiplied by the ratio of the quantum partition function and the classical partition function for a harmonic oscillator:

$$\begin{aligned} Q_{\text{HR}_{\text{semi}}} &= Q_{\text{HR}_{\text{classical}}}\frac{Q_{\text{RRHO}_{\text{quantum}}}}{Q_{\text{RRHO}_{\text{classical}}}} \\ &= \left(\frac{\pi k_B T}{\sigma^2 B}\right)^2 e^{-V_0/2k_BT} I_0\left(\frac{V_0}{2k_BT}\right)\frac{\frac{e^{-\nu/2k_BT}}{1-e^{-\nu/k_BT}}}{\frac{k_BT}{\nu}} \end{aligned} \tag{17.7}$$

where the frequency ν is the harmonic oscillator frequency approximation to the bottom of the potential well, and k_B is Boltzmann's constant. According to Knyazev [Knyazev1994] (page 156), the inverse Laplace

transform for the hindered rotor partition function is:

$$\rho_{\mathrm{HR}_{\mathrm{semi}}}(E) = \begin{cases} \dfrac{2\mathbf{K}\left(\sqrt{E/V_0}\right)}{\pi\sigma\sqrt{B_r V_0}} & \text{for } E < V_0 \\ \dfrac{2\mathbf{K}\left(\sqrt{V_0/E}\right)}{\pi\sigma\sqrt{B_r E}} & \text{for } E > V_0 \end{cases}$$

where $\mathbf{K}$ is the complete elliptical integral of the first kind. The equation above is calculated for each hindered rotor. If a molecule has more than one hindered rotor, then the total density of states for hindered rotors is obtained by successively applying the convolution integral. The density of states for internal rotors and active external rotation is calculated by convoluting the density of states for the active K-rotor with the density of states for all the hindered rotors:

$$\rho_{r,HR_{\mathrm{total}}}(E) = \int_0^E \rho_r(E-x)\,\rho_{\mathrm{HR}_{\mathrm{total}}}(x)\,dx \tag{17.8}$$

Finally, following the method of Astholz [Astholz1979] [Gilbert1996], we initialize the Beyer-Swinehart algorithm [Beyer1973] with the K-rotor/hindered-rotor density of states. This algorithm convolutes this with the vibrational modes. The result is the density of states for all the active degrees of freedom, $\rho(E)$.

17.3.4 Frequencies and Barrier Heights

To summarize everything up to this point: Our objective is to calculate pressure dependent rate constants for chemically-activated reactions. To calculate these rate constants, we must solve the master equation. In order to solve this system of equations, we need the micro-canonical rate constants for the association, isomerization, and dissociation channels. These rate constants, in turn, require the density of states for the adducts. Last but not least, the density of states requires detailed information about the rigid-rotor harmonic-oscillator frequencies as well as hindered rotors parameters. Normally, these physical parameters would be determined from a 3D model of the molecule.

Unfortunately, RMG does not have access to 3D structure models at this time (although we are working on it). The current version of RMG represents each molecule solely by its connectivity. The connectivity diagram for each species is used to calculate the heat capacity at seven temperatures (300, 400, 500, 600, 800, 1000, 1500 K), based upon the group additivity theories developed by Benson [Benson1976] (page 156).

Since the heat capacity is a function of the vibrational and rotational modes, it is possible, in theory, to determine the frequencies and barrier heights from the heat capacity. In practice, however, this is almost never possible, since the number of unknown vibrational/torsional parameters ($3N - 6$) will most likely exceed the number of C_p values (7). Consequently, some approximations must be made. The remainder of this section describes the subroutine that RMG uses to estimate the unknown parameters.

RMG uses a two-step process to estimate the vibrational frequencies and hindered rotor barrier heights. In the first step, the subroutine will examine the structure of the molecule to determine the types of groups present; it will then use this information to predict most of the vibrational frequencies. In the second step, the subroutine examines the heat capacity as a function of temperature to determine the barriers to internal rotation and the remaining vibrational frequencies.

Estimating the vibrational frequencies based on the group type

What is meant by "group type" is unique to this subroutine. In principle, however, it is similar to functional groups. Thus, methyl groups are one group type, as are ethers, peroxides, branched and straight-chain alkyl backbones, etc. For each type of group, we have compiled a list of corresponding vibrational modes. To take methyl groups (R-CH3) as an example, there are nine frequencies that are common to methyl groups: 3 C-H stretches, 2 R-C-H bending modes, 2 R-C-H rocking modes, 1 R-C stretch, and 1 umbrella mode. Each of these modes has an upper and lower limit (for example, the C-H stretch in methyl groups is typically confined between 2750 and 2850 cm^{-1}).

For a non-linear molecule with N atoms and R internal rotors, there will be 3N-6-R vibrational modes. The objective of the first part of the subroutine is to determine as many of these RRHO frequencies as possible, as accurately as possible, without determining too many. If the code determines fewer than 3N-6-R modes, that is ok, because they will be estimated in the second section of the code. If, however, the code determines more than 3N-6-R modes, there is a serious problem. To avoid this complication, we have limited the number of modes attributed to each group type so that this situation can never arise.

For each group type, we count both the number of atoms in that group as well as the number of single bonds between heavy atoms in that group (since each of these bonds may contribute one internal rotor). Thus, we limit the number of modes for each group to 3 $\times$ (Number of atoms in group) - (number of single bonds between heavy atoms in group) - 6. This approach guarantees that the number of RRHO modes is less than 3N-6-R. By using this method, the first section of the code will predict almost all of the RRHO modes.

Estimating the vibrational frequencies based on the heat capacity

The second part of the subroutine uses the heat capacity to determine both the remaining unknown vibrational modes as well as the barrier heights of any hindered internal rotors. Using Benson-type group additivity formulas, RMG will calculate the heat capacity for the molecule at seven temperatures. At each temperature, the heat capacity can be separated into the contributions from external translation, external rotation, rigid-rotor harmonic-oscillator frequencies (RRHO), and hindered internal rotation (HR):

$$C_V^{\text{total}} = C_V^{\text{translation}} + C_V^{\text{rotation}} + \sum_{j=1}^{3N-6-N_{\text{rotors}}} C_V^{\text{RRHO}} + \sum_{j=1}^{N_{\text{rotors}}} C_V^{\text{HR}}$$

The first part of the subroutine should estimate some of the RRHO vibrational frequencies, call it $N_{estimated}$. The contribution of the estimated frequencies, as well as the contributions from the external translation and rotation, are subtracted from the total heat capacity. What remains is the heat capacity due to the unknown degrees of freedom (udof):

$$\begin{aligned} C_V^{\text{udof}} &= C_V^{\text{total}} - C_V^{\text{translation}} - C_V^{\text{rotation}} - \sum_{j=1}^{N_{\text{estimated}}} C_V^{\text{RRHO}} \\ &= \sum_{j=1}^{N_{\text{remaining}}} C_V^{\text{RRHO}} + \sum_{j=1}^{N_{\text{rotors}}} C_V^{\text{HR}} \end{aligned} \tag{17.9}$$

where $N_{\text{remaining}} \equiv 3N - 6 - N_{\text{rotors}} - N_{\text{estimated}}$. The formula for the heat capacity for the i^{th} rigid-rotor harmonic-oscillator frequency can be found in any text on statistical mechanics [McQuarrie1973] (page 156)

[Hill1986] (page 156):

$$\frac{C_V^{\text{RRHO}}(T;\nu_i)}{R} = e^{\nu_i/k_BT}\left(\frac{\nu_i/k_BT}{e^{\nu_i/k_BT}-1}\right)^2 \tag{17.10}$$

where ν_i is the vibrational frequency for the i^{th} mode.

As we saw in the *Density of States Estimation* (page 131) section, the approximation used for the heat capacity of a hindered internal rotor is more complicated. Recall the semi-classical partition function for a hindered rotor:

$$\begin{aligned} Q_{\text{HR}_{\text{semi}}} &= \left(\frac{\pi k_BT}{\sigma^2 B}\right)^2 e^{-V_0/2k_BT} I_0\left(\frac{V_0}{2k_BT}\right)^{\frac{e^{-\nu/2k_BT}}{1-e^{-\nu/k_BT}}}_{\frac{k_BT}{\nu}} \\ &= \left(\frac{V_0\pi}{2\sigma^2 B}\right)^{1/2} u^{1/2} e^{-u} I_0(u)\,\frac{e^{-\alpha u}}{1-e^{-2\alpha u}} 2\alpha \end{aligned}$$

where $u \equiv V_0/2k_BT$, and $\alpha \equiv \nu/V_0$. The frequency ν is the harmonic oscillator frequency approximation to the bottom of the potential well. The heat capacity for this partition function is:

$$\begin{aligned} \frac{C_V^{\text{HR}}(T;V_0,\nu)}{R} &= \frac{d}{dT}\left(T^2\frac{d\ln Q_{\text{HR}}}{dT}\right) \\ &= u^2\frac{d^2\ln Q_{\text{HR}}}{du^2} \\ &= -\frac{1}{2} + u\left[u - \frac{I_1(u)}{I_0(u)} - u\left(\frac{I_1(u)}{I_0(u)}\right)^2\right] + \left(\frac{2u\alpha e^{-\alpha u}}{1-e^{-2\alpha u}}\right)^2 \\ &= -\frac{1}{2} + \frac{V_0}{2k_BT}\left[\frac{V_0}{2k_BT} - \frac{I_1\left(\frac{V_0}{2k_BT}\right)}{I_0\left(\frac{V_0}{2k_BT}\right)} - \frac{V_0}{2k_BT}\left(\frac{I_1\left(\frac{V_0}{2k_BT}\right)}{I_0\left(\frac{V_0}{2k_BT}\right)}\right)^2\right] + \left(\frac{\frac{\nu}{k_BT}e^{-\frac{\nu}{2k_BT}}}{1-e^{-\frac{\nu}{k_BT}}}\right)^2 \end{aligned} \tag{17.11}$$

Fortunately, both the symmetry number σ and the reduced moment of inertia I_r do no appear in the expression for the heat capacity. What remains is the barrier height, V_0, and the frequency, ν. The heat capacity for the unknown degrees of freedom may now be written as:

$$C_V^{\text{udof}}(T) = \sum_{i=1}^{N_{\text{remaining}}} C_V^{\text{RRHO}}(T;\nu_i) + \sum_{j=1}^{N_{\text{rotors}}} C_V^{\text{HR}}(T;V_{0,j},\nu_j)$$

Unfortunately, even if the first part of the subroutine is successful in determining most of the vibrational modes, it is unlikely that it will determine all of them. Furthermore, since there are only seven heat capacity data, it is unlikely that all of the remaining parameters can be determined by the second part of the subroutine. In many cases, either the rigid-rotor harmonic-oscillator frequencies or the hindered rotors (and in some cases both) must be lumped into a single pseudo-frequency or pseudo hindered rotor, respectively:

$$\begin{aligned} \sum_{i=1}^{N} C_V^{\text{RRHO}}(T;\nu_i) &\rightarrow N C_V^{\text{RRHO}}(T;\nu) \\ \sum_{j=1}^{N} C_V^{\text{HR}}(T;V_{0,j},\nu_j) &\rightarrow N C_V^{\text{HR}}(T;V_0,\nu) \end{aligned}$$

Depending upon the exact number of unknown frequencies and rotors, the code will call 26 possible fitting subroutines. In order to make the resulting parameters as realistic as possible, there must be physically meaningful constraints. Thus, the RRHO frequencies are bound between 400 and 4000 cm^{-1}. The HR parameters are less constrained: the low-temperature frequency is bound between 40 and 600 cm^{-1}, and the barrier height is bound between 10 and 10000 cm^{-1}. The fitting procedure is a bounded, constrained least squares fit for a system of highly nonlinear equations, which is a difficult problem to solve. To accomplish this fitting, we use the publicly available code DQED [DQED] (page 156) [DQED90] (page 156).

17.4 Automated Time Stepping

A means of "automated time stepping" in RMG was implemented in order to eliminate the need for the user to specify times/conversions for mechanism validity testing. The "AUTO" option implementing this "automated time stepping" was successfully integrated into RMG starting in version 3.00. Several tests have been carried out with the new "AUTO" option. Results were promising, suggesting that the option functions as intended without being unreasonably computationally expensive. The "AUTO" option offers a more rigorous implementation of RMG's rate-based mechanism generation algorithm while allowing straightforward generation of reaction mechanisms that encompass chemistry at all time-scales of interest.

17.4.1 Motivation

Originally, the implementation of RMG's reaction mechanism generation algorithm required the user to specify times or conversions at which the program will estimate "edge" reaction fluxes and decide whether to include a new species in the "core" reaction model. The choice of which conversions or times to specify is, to a certain extent, arbitrary, and could have a significant impact on the model generated by the program. This arbitrariness can be effectively eliminated by keeping track of the "edge" reaction flux "continuously" within the ODE solver (analogously to how conversion is handled) and terminating ODE solver "integration" when the threshold reaction flux is reached (whereupon a species would be moved from the "edge" to the "core" as in the current RMG implementation). This new method is here referred to as "automated time stepping". Implementation of such a method would reduce the number of adjustable parameters that the user must tweak during mechanism generation and provide a more rigorous use of the mechanism generation algorithm. At the same time, such a method would allow straightforward generation of mechanisms that encompass chemistry at all time-scales of interest.

17.4.2 Background

The reaction mechanism is divided into a "core" and an "edge". The "core" is tracked by integrating the appropriate equations using an ordinary differential equation (ODE) solver. The "edge" includes species that are products of reactions among the "core" species. Periodically during the course of simulating species concentrations using the ODE solver, mechanism validity is tested. Mechanism validity criteria used by RMG are shown in the equations below.

$$\frac{dC_j}{dt} < tol \cdot R_{char} \forall j \in \text{edge species} \tag{17.12}$$

$$\text{with } R_{char} = \left[\sum_i \left(\frac{dC_i}{dt} \right)^2 \right]^{\frac{1}{2}} i \in \text{core species} \tag{17.13}$$

In these criteria, tol is a tolerance specified by the user. If any of these criteria are not valid, the edge species with the highest flux is added to the mechanism "core" and the ODE solver restarts simulation from t = 0. In the original implementation, the times or conversions at which model validity is tested are specified by the user. The ODE solvers used by RMG are FORTRAN programs (DASSL or DASPK), while RMG itself is written in Java.

17.4.3 Implementation

In order to achieve the desired behavior for "automated time stepping", modifications of the type illustrated in Figure 1 were required.

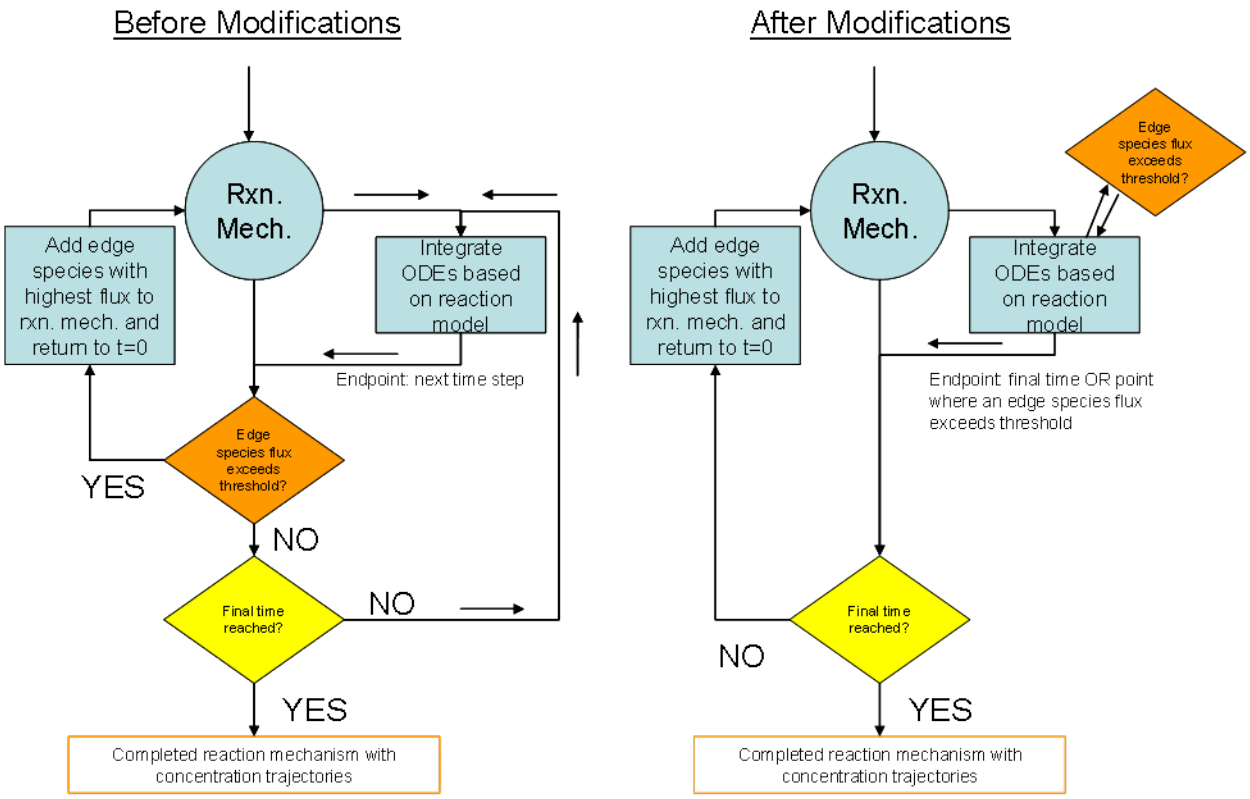


Figure 17.2: Figure 1. Modifications to implement "automated time stepping".

As the figure shows, a significant required change was the modification of the ODE solver to have access to edge species flux information at each ODE solver time step. Thus, much of the modification effort involved implementing a scheme for passing edge reaction information to the ODE solver. The implementation ended up requiring changes to the Java classes ReactionModelGenerator and JDASSL and the FORTRAN file call_dassl.f (the modified version was renamed to call_dasslAUTO.f90 to distinguish from the original version). Also, a minor change to the Java class ReactionTimeTT was made. An example of an ODE solver input file for "automated time stepping" is shown in Figure 2, below. Differences from "normal" operation are highlighted with boxes.

The first box in the above figure highlights an integer flag signalling to the FORTRAN code that "automated time stepping" is desired (as opposed to "normal" operation"). The next box highlights the section

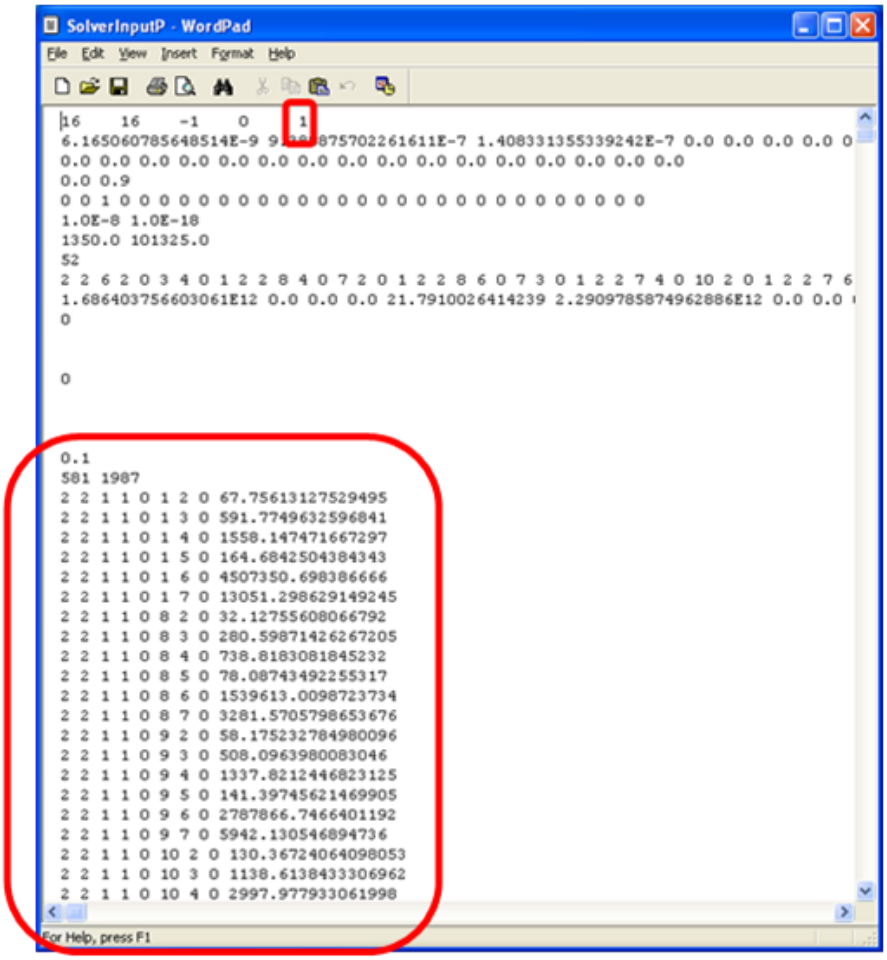


Figure 17.3: Figure 2. ODE solver input file example for “automated time stepping”. Boxes highlight differences from the “normal” version.

containing the information needed by the ODE solver to check mechanism validity at each step. In particular, this includes the user-specified tolerance, the number of edge species/reactions, and a line for each edge reaction containing the number of reactants, the number of products, ID numbers corresponding to reactants and products, and finally the rate coefficient, k, for the reaction. It should be noted that in the current implementation, the rate coefficient is considered to be a constant, as additional information regarding temperature/pressure dependence is not passed to solver. Thus, the "automated time stepping" code (along with other preexisting DASSL code) would need to be modified to handle non-isothermal cases (as well as non-isobaric pressure-dependent cases). In order to reproduce existing validity testing for cases with pressure-dependence, special considerations are necessary. In particular, each pressure-dependent network is considered as a "pseudo-species" with an associated "pseudo-reaction" using the kleak associated with the network as the rate coefficient. Monitoring flux to these "pseudo-species" thus allows us to consider whether a pressure-dependent network needs to be enlarged, analogously to how monitoring of species flux allows us to consider whether a species needs to be added to the "core" reaction mechanism.

17.4.4 Usage notes

The modified code still allows for evaluating flux at user-defined time/conversion points, and the use of "automated time stepping" is invoked by replacing the user-defined time/conversion points with the keyword "AUTO". Thus, the use of "automated time stepping" will be referred to as the "AUTO" method in subsequent discussion. The modified code has been designed to work for both pressure-dependent and non-pressure-dependent cases; also, the code allows automatic stepping with either a conversion or a reaction time as the user-specified termination criterion.

17.4.5 Results

The new option of "AUTO" time-stepping was tested for several cases to verify that the option was working as desired and to investigate the effects on mechanism generation time and on the resulting mechanism. Note: Throughout this section, "1,3-hexadiene" refers to a system involving methane doped with 1,3-hexadiene (in particular, the system defined by the condition files distributed with RMG 2.00 (March 2007)). A brief summary of findings follows.

Effects on Generated Mechanism

One would expect that mechanisms generated using "AUTO" method would be more rigorously valid than those generated using arbitrary user-specified time/conversion steps. In the case of the 1,3-hexadiene system, mechanisms generated using the "AUTO" method were compared to mechanisms generated using evenly-spaced conversion steps. The results are shown in Table 1, below.

Table 17.1: Table 1. Size of mechanisms generated using the "AUTO" method and varying numbers of evenly-spaced conversion steps. In all cases, the goal conversion is 0.9 and the tolerance is 0.05.

# of conversion steps	Core reactions	Core species
1	308	31
2	56	17
3	55	16
4	55	16
6	55	16
9	48	15
Auto	48	15

It should be noted that the mechanism generated with the "AUTO" method and the mechanism generated using nine evenly-spaced conversion steps appear to be essentially identical for this system. Thus, in this case, we approach "AUTO" method behavior as we specify a finer "grid" for conversion steps, as one might intuitively expect. It is also interesting to note that in the test case considered, the "AUTO" method produced a smaller mechanism than cases with a small number of specified conversion steps. Thus, in this case, the mechanism generated by the "AUTO" method (and cases with a sufficiently fine grid of conversion steps) would not be expected to be more accurate than the mechanism generated without any conversion steps (assuming the kinetics and thermochemistry of the extra species and reactions is accurate). However, the time required to generate the "AUTO" mechanism is significantly shorter, due to the smaller size of the mechanism. Furthermore, one can imagine that mechanisms of similar size to the largest mechanism reported above could be generated by using tighter tolerances and the "AUTO" method, and one could be more confident in their validity. In summary, although it produces a larger mechanism, the use of a low number of conversion steps is not "better" than the use of a high number of conversion steps or the "AUTO" method.

Timing Studies

Tests suggest that the time required to generate a mechanism with the "AUTO" method is virtually the same as the time required to generate the same mechanism with conventional use of conversion steps. However, the computational cost associated with tracking edge-species is significant compared to the cost of solving the ODEs, with the "AUTO" cases requiring a longer amount of time by a factor of two to eight (for the same number of ODE time steps). Three phenomena are proposed to explain this "discrepancy". First, one would expect that in the "AUTO" case, the total time that we must integrate over during mechanism generation will be shorter, in general; that is, the "AUTO" method avoids "wasted" ODE time steps associated with "overshooting" the point where threshold flux has been reached. Second, the time associated with ODE solving is small in comparison to the total time requirement (< 10% of the total time for mechanism generation). Third, and perhaps most significant, there are fewer calls to the ODE solver when the "AUTO" method is used. In the case of the "AUTO" method, each call to the ODE solver is associated with a modification to the reaction mechanism (with the exception of the final integration). However, in the case of the "normal" method, there are extra calls to the ODE solver, as the integration is often stopped despite the mechanism being valid (i.e. no edge species fluxes exceed the threshold). Thus, in order to generate the same mechanism, the "normal" method requires a larger number of calls to the ODE solver, which, in turn is associated with greater overhead such as reading and writing ODE solver input and output files. Timing studies for the

1,3-hexadiene test case suggest that this overhead can be significant. (Note that the use of eighteen rather than nine conversions exacerbates the time cost, in accord with expectations based on the aforementioned overhead considerations.) The contribution of this overhead to FORTRAN time requirements is relatively small, but noticeable, while the contribution from the overhead on the Java side appears to be more significant. In fact, in this case, the total ODE solver time-requirement is noticeably lower in the "AUTO" case (relative to the "normal" cases). Thus, in this case, it appears that the added overhead associated with extra calls to the ODE solver in the "normal" method has outweighed the extra cost associated with tracking the flux at each time step within the ODE solver, using the "AUTO" method.

17.4.6 Summary

The "AUTO" method was implemented and tested during development of RMG 3.00 and provides the option of "automated time stepping". Even so, timing studies suggest that tracking the edge flux "continuously" in the ODE solver is associated with a significant time penalty. However, various mitigating factors appear to reduce the impact of this time penalty on overall mechanism generation time when the "AUTO" method is used. Thus, the "AUTO" method appears to be an appealing alternative to user-specification of time/conversion steps.

CHAPTER

EIGHTEEN

FAQ (FREQUENTLY-ASKED QUESTIONS)

See also the *Troubleshooting and Known Issues* (page 147) section.

18.1 Questions

18.1.1 How do I represent molecular oxygen?

Use the ground electronic state of molecular oxygen, which contains a single bond and a diradical. The adjacency list should be

```
O2 (mol/cm3) 0.1
1 O 1 {2,S}
2 O 1 {1,S}
```

For more information, please see *Representing oxygen* (page 107).

18.1.2 My RMG run runs out of memory. How should I proceed?

This type of error will probably look something like `Exception in thread "main" java.lang.OutOfMemoryError: Java heap space`. Species take up a large amount of memory, and when the combined edge plus core gets into the range of tens of thousands of species, the run will typically crash even with around 1500 MB of RAM. If you haven't already done so, we recommend allocating as much memory as possible when executing the RMG job (for example, using the option -Xmx1600m will allocate 1600 MB of memory. See the section on *running RMG* (page 79)).

Note: The way Java creates subprocesses (e.g. for running the ODE solver DASSL or Pressure-Dependence solver FAME) it allocates the process with the same amount of memory as the main program. If you had you heap set to 1GB then it will allocate another 1GB for the sub-process to run. Therefore you shouldn't set the `-Xmx` amount to more than half of your available memory, or you will run out of memory when starting one of the subprocesses.

If this still fails, there are several additional options:

1. Use the *pruning* (page 43) feature introduced in RMG 3.2.
2. Use tighter bounds for MaxCarbonNumber, MaxRadicalNumber, etc. as appropriate for your system. By reducing these values, RMG will generate fewer edge species. The idea is to intelligently choose the bounds to exclude species that shouldn't be relevant for your particular system.
3. Relax the error tolerance (i.e. make it larger). RMG keeps expanding the reaction mechanism until the flux to 'edge' species is below the tolerance you specify. Increasing this tolerance means RMG will build a smaller mechanism (though it may be missing some important reaction pathways). For more details see *Rate-Based Model Enlarger* (page 126) and the reference on that page.

18.1.3 RMG has been running for multiple days and hasn't finished. Is there any way to make it run faster?

Getting RMG to converge in a reasonable time, especially for large starting species, can be tricky. Depending on the options specified, the RMG developers have seen mechanism generation runs as long as ~10 days. During this time, RMG is running hundreds of reaction model simulations and considering tens of thousands of species, and perhaps over a million possible reactions. As such, it can be inconvenient to have it running for a long time on your personal single-core machine or laptop. So, if you have access to a computing cluster where you can run RMG, that can ease the burden. Some "tricks" that could be helpful in reducing time requirements for model generation (in no particular order):

1. Relax the error tolerance (i.e. make it larger). RMG keeps expanding the reaction mechanism until the flux to 'edge' species is below the tolerance you specify. Increasing this tolerance means RMG will stop sooner; it will take less time, but the model will be smaller and may be missing some important reaction pathways. For more details see *Rate-Based Model Enlarger* (page 126) and the reference on that page.
2. Setting the maximum number of carbons (or oxygens, heavy atoms, etc.) in a species, if you're focusing on the species decomposition.
3. Run RMG for a single temperature / pressure (TemperatureModel, PressureModel).
4. If you are running RMG with pressure dependence, consider trying the less accurate (but faster) ModifiedStrongCollision option.
5. If you are using a Seed Mechanism, set the GenerateReactions field to off.

18.1.4 How do I use the DASPK dynamic solver?

RMG uses DASPK in a manner that requires a proprietary library known as DAEPACK, and unfortunately, we cannot distribute this library. If you acquire this library (for example, possibly through the Jacobian program from Numerica) we will be able to help you compile the Linux version of daspkAUTO.exe, but compiling in Windows has been tricky in our experience.

Hopefully, DASSL will meet your needs as it should provide all the functionality of DASPK, with the exception of sensitivity analysis (which you could do in Chemkin with the final model).

18.1.5 During the RMG run, I see the warning: "Frankie exceeded maximum number of iterations...". Is this normal?

This occurs when the optimization algorithm for the frequency estimation code "Frankie" fails to converge. In these cases, the frequencies (used to calculate density of states in master equation calculations to account for pressure dependence) may not be as accurate as they could be. It is typical for some fraction of Frankie jobs to fail in this fashion.

18.1.6 During the RMG run, I see the warning: "fitted Er < 0 ...". Is this normal?

This occurs when the fitted activation energy for reverse reaction rate coefficients (based on equilibrium considerations) turns out to be negative. In these cases, the activation energy for the reverse reaction is adjusted to zero, with corresponding adjustment to the A-factor to ensure accurate equilibrium constant is maintained. It is typical for some fraction of the large number of reactions considered to produce this output.

18.1.7 I encounter a NegativeConcentrationException, causing the RMG run to fail. How should I proceed?

The error message will read something like `Exception in thread "main" jing.rxnSys.NegativeConcentrationException...`. The issue here is that the ODE simulator reached a slightly negative concentration, which RMG consequently interpreted as erroneous and stopped. It is basically a numerical artifact, as the equations should not allow a negative concentration. There are two options for dealing with this:

1. Play around with the value for Atol in the *Dynamic Simulator* (page 42) block in your condition file...for example, maybe try tightening it (i.e. reduce the value) by a few orders of magnitude.
2. Specify the non-negativity option (introduced in version 3.2). (This could cause other solver errors, so use this as a last-resort.) To use this option with the DASSL solver, specify non-negative in the condition.txt input file as follows: "DynamicSimulator: DASSL: non-negative".

18.1.8 During the RMG run, I see the warning: "Negative Steady State populations encountered during reservoir state method, falling back to Modified Strong Collision". Is this normal?

This is a normal numerical error in the matrix algebra. It occurs sometimes with the reservoir state method due to the larger size of the matrices. RMG recovers from the error by trying the Modified Strong Collision method.

CHAPTER

NINETEEN

TROUBLESHOOTING AND KNOWN ISSUES

19.1 Getting help

If the tips on this page don't solve your problem, there are several ways you can ask for help, and we encourage you to do so. If we don't hear what our users have trouble with, we can't improve the software.

Users mailing list There is an RMG-Users mailing list (https://lists.sourceforge.net/lists/listinfo/rmg-users) hosted by soureforge that you are welcome to join and/or email.

Developers mailing list The developers can be reached quickly by emailing *rmg_dev@mit.edu*. We welcome your questions.

Issue Tracker The developers use an issue tracker (https://github.com/GreenGroup/RMG-Java/issues/) at https://github.com/GreenGroup/RMG-Java/issues/ to keep track of issues. Feel free to open an issue there. If searching for issues other people have had, be sure to check the closed issues (https://github.com/GreenGroup/RMG-Java/issues/closed) too.

19.2 Known issues

19.2.1 Results differ from machine to machine.

The ODE solver (DASSL) will return slightly different results depending on the Fortran compiler, the compilation options, the machine architecture, etc. This is not a mistake, as all the results will be within the requested precision (*ATOL and RTOL* (page 44)). However, in instances where two reaction fluxes have similar or identical rates, the small difference can lead to a different decision being made regarding which species to add to the core or which pressure-dependent network to explore next.

19.2.2 Results differ from run to run.

If you run the same input file twice on the same computer, the resulting chemistry should be the same. However, when a species with several isomers is created there is no guarantee that they will be numbered

the same way. Therefore the species numbers (and hence names) will differ from run to run. This will make the log files and chemkin files look different.

19.2.3 Job freezes for a long time while running Fame.

Sometimes a pressure-dependent network calculation in Fame can have a very large output. When this output stretches to tens of megabytes, sometimes it can block the pipe through which the Fame program (written in Fortran) and the RMG program (in Java) communicate. When the computer is running low on memory this can be especially troublesome. On some computers the job has been known to freeze for tens of hours. The symptom is the job pauses, although the task manager of the computer suggests not a lot is happening. The only sure way to avoid this is to build a smaller model: use a higher termination criterion.

19.2.4 RMG does not recognize kinetics from multiple reaction family templates for the same structure

A "structure" is defined in RMG by a list of the reaction's reactants and products, e.g., A <=> B or A <=> B + C. RMG can predict different pathways for the same structure, i.e. there are multiple ways for "A" to react to "B". Some examples

1. **n-butyl (.CH2-CH2-CH2-CH3) <=> iso-butyl (CH3-.CH-CH2-CH3)** In this example, RMG will find two pathways between reactant and product: the three- and four-membered cyclic transition state. RMG estimates kinetics for each pathway and considers both in its mechanism generation algorithm.

   This materializes in the RMG output (particularly, the `chem.inp` file) in the following manner (for pressure-dependent simulations):

   ```
   C4H9-1(1)(+m)=C4H9-2(2)(+m)                                1.0E0 0.0 0.0   !NetReacti
   !intra_H_migration exact: [ Others-R3H_SS , C_rad_out_2H , Cs_H_out_H/NonDeC ]
   ```

   For pressure-independent simulations, the `chem.inp` will represent the multiple pathways in this manner:

   ```
   C4H9-1(1)=C4H9-2(2)                                  1.180e+10   0.82    35.10  !i
   DUP
   C4H9-1(1)=C4H9-2(2)                                  1.938e+10   0.89    35.80  !i
   DUP
   ```

2. **iso-propyl (CH3-.CH-CH3) + O2 <=> HO2 + propene** In this example, RMG will find the chemical-activation route (i-C3H7 + O2 <–> .OOCH(CH3)2 <–> HOOCH(.CH2)(CH3) <–> HO2 + C3H6) and the pressure-independent route (via the Disproportionation reaction family template). These distinct kinetics materialize in the `chem.inp` (for pressure-dependent simulations) in the following manner:

   ```
   iC3H7(1)+O2(2)(+m)=C3H6(4)+HO2(3)(+m)                1.0E0 0.0 0.0   !NetReacti
   TCHEB / 300.0 2000.0 /  PCHEB / 0.009869232667160128 98.69232667160128 /
   CHEB / 4         4 /
   CHEB / 1.1842000e+01 -6.3021000e-01 -1.3307000e-01 -1.8047000e-03 /
   CHEB / 6.3958000e-01 7.1487000e-01 1.1880000e-01 -1.4505000e-02 /
   CHEB / 1.0687000e-01 -5.1396000e-03 3.1099000e-02 1.5790000e-02 /
   ```

```
CHEB / 1.7064000e-02 -7.5091000e-02 -8.5661000e-03 2.2822000e-03 /

O2(2)+iC3H7(1)=HO2(3)+C3H6(4)                    1.270e+11   0.00    0.00   !D
```

3. **C2H4 + .CH2-CH2-OH <=> .CH2-CH2-CH2-CH2-OH** In this example, RMG can match this structure to two different reaction family templates: R_Addition_MultipleBond (where the C2H4 corresponds to .**C**H2-**C**H2-CH2-CH2-OH in the product) and 1,3_Insertion_ROR (where the C2H4 corresponds to .CH2-**C**H2-**C**H2-CH2-OH in the product). However, since the kinetics arise from different reaction family templates, RMG will currently only utilize the first set found in its mechanism generation algorithm (the other will be treated as a unwanted duplicate reaction and discarded).

   In a normal RMG run (with all reaction family templates turned on), the R_Addition_MultipleBond reaction family template will be found:

   ```
   C2H4(1)+SPC(2)=SPC(3)                            3.980e+03   2.44    5.37   !R
   ```

   Turning off the R_Addition_MultipleBond reaction family template will produce this alternative result:

   ```
   C2H4(1)+SPC(2)=SPC(3)                            1.304e+03   2.92    44.63  !1
   ```

   In this case, RMG (luckily) utilizes the faster of the two kinetics. However, this is **NOT** guaranteed: RMG uses the first kinetics instance found and ignores all other kinetics from different reaction family templates for the same structure.

While examples #1 and #2 are currently handled correctly in RMG, example #3 is not (only one instance of the B + C <=> A will be used in the simulation and reported in the output files).

19.2.5 RMG "Primary Kinetic Libraries" and "Reaction Libraries" do not handle irreversible reactions properly

If a reaction is specified as irreversible in a "Primary Kinetic Library" or "Reaction Library," RMG will read in the reaction's structure and kinetics properly. For example, suppose the reaction C2H5 => C2H4 + H were supplied in a "Primary Kinetic Library". If a user supplied an input file to RMG with C2H5 in the condition file, RMG would run C2H5 against all RMG-defined reaction family templates. One of these templates would be R_Addition_MultipleBond; the reverse reaction family template (Beta_Scission) of which would find C2H5 = C2H4 + H. Thus, RMG would assign the kinetics supplied by the user in the "Primary Kinetic Library" to this reaction, instead of estimating it by traversing the kinetics tree.

However, the RMG reaction family template "Beta_Scission" is defined as reversible and thus RMG will also construct the reaction C2H4 + H => C2H5; the kinetics are computed through equilibrium. As of now, RMG cannot recognize that the user requested the reaction to be irreversible, so the mechanism generation algorithm will consider the reaction to be reversible. All RMG output files will thus report this reaction as reversible.

The user is advised **NOT** to supply irreversible reactions in any "Primary Kinetic Library" or any "Reaction Library".

19.3 Troubleshooting

19.3.1 Final attempt (#36) on species ____ failed.

With QMTP: if after "Final attempt" it still fails, and reports the chemgraph of the species that failed, take this chemgraph and either provide thermochemistry for it in a primaryThermoLibrary, or else add it to your ForbiddenStructures section of your condition file or database.

19.3.2 Error with the ODE solver

You may see this at the end of your log file:

```
Reaction System 3 has not reached its termination criterion
although it seems to be valid (complete), so it was not interrupted for being invalid.
This probably means there was an error with the ODE solver, and we risk entering an endless
Stopping.
```

If you scroll further up you will probably find that at least one of your reaction system simulations stopped with an error report from the ODE solver. This can happen if the system of differential equations becomes too stiff, and/or your error tolerances (ATOL and RTOL) for the solver are inappropriate. Try modifying them (either up or down), or use a looser termination criterion, or less extreme reaction system conditions. If RMG has nothing to do (all the systems have either converged or failed) then it would try running the simulations again without expanding the model, with the same result; to avoid an infinite loop, it stops with this message. The good news is, this only causes RMG to stop running if all the reaction systems that solved successfully have already converged and met their termination criteria.

19.4 Making valid CHEMKIN files

To create valid CHEMKIN files, ensure that any 3rd-body collider molecules used in your Seed Mechanisms or Reaction Libraries are either declared in their species.txt files (if they are reactive) or in the *Inert Gases* (page 38) section of the input file. (i.e. it's a good idea to list all of N2, Ne, He and Ar, even if their concentrations are 0).

The CHEMKIN input file `chem.inp` generated with the *Verbose* (page 48) option turned "on" may have a comment that spans hundreds of characters. These verbose comments may cause the CHEMKIN interpreter to fail when running the Pre-Processor.

When running RMG on Windows it is possible that the `chem.inp` file will contain a mixture of linux and windows line endings (http://en.wikipedia.org/wiki/Newline). This can be a problem, for example if you want to read the chemkin file using Cantera (http://code.google.com/p/cantera/). Convert the line endings using a program like dos2unix.

CHAPTER

TWENTY

CREDITS

RMG is based upon work supported by the Department of Energy, Office of Basic Energy Sciences through grant DE-FG02-98ER14914 and by the National Science Foundation under Grants No. 0312359 and 0535604.

Project Supervisor:

- Prof. William H. Green (whgreen@mit.edu (whgreen@mit.edu))

Current Developers: (rmg_dev@mit.edu (rmg_dev@mit.edu))

- Joshua W. Allen
- Dr. Beat A. Buesser
- Caleb A. Class
- Connie Gao
- Amrit Jalan
- Dr. Murat Keceli
- Shamel S. Merchant
- Dr. Yury Suleymanov
- Prof. Richard H. West
- Nathan Yee

Original Developer:

- Dr. Jing Song

Previous Developers:

- Dr. Gregory R. Magoon
- Jeffrey D. Mo
- Dr. Michael R. Harper
- Dr. C. Franklin Goldsmith
- Dr. Sandeep Sharma

- Dr. Robert W. Ashcraft
- Dr. Gregory J. Beran
- Dr. David M. Matheu
- Dr. Sumathy Raman
- Dr. Joanna Yu
- Sarah Petway
- Dr. Paul E. Yelvington
- Dr. John Wen
- Andrew Wong
- Dr. Hsi-Wu Wong
- Prof. Kevin M. Van Geem

RMGVE Developer:

- John Robotham

Special thanks for helpful error reports:

- Rajesh D. Parmar
- Dr. Seyed-Abdolreza Seyed-Reihani
- Dr. Yu Shi
- Dr. Yushi Suzuki
- Aaron Vandeputte
- Dr. Stijn Vranckx
- Dr. Steven Zabarnick

CHAPTER

TWENTYONE

HOW TO CITE

To cite RMG, please refer to this website: William H. Green, Joshua W. Allen, Beat A. Buesser, Robert W. Ashcraft, Gregory J. Beran, Caleb A. Class, Connie Gao, C. Franklin Goldsmith, Michael R. Harper, Amrit Jalan, Murat Keceli, Gregory R. Magoon, David M. Matheu, Shamel S. Merchant, Jeffrey D. Mo, Sarah Petway, Sumathy Raman, Sandeep Sharma, Jing Song, Yury Suleymanov, Kevin M. Van Geem, John Wen, Richard H. West, Andrew Wong, Hsi-Wu Wong, Paul E. Yelvington, Nathan Yee, Joanna Yu; "RMG - Reaction Mechanism Generator v4.0.1", 2013, http://rmg.sourceforge.net/

BIBLIOGRAPHY

[GRIMech3.0] Gregory P. Smith, David M. Golden, Michael Frenklach, Nigel W. Moriarty, Boris Eiteneer, Mikhail Goldenberg, C. Thomas Bowman, Ronald K. Hanson, Soonho Song, William C. Gardiner, Jr., Vitali V. Lissianski, and Zhiwei Qin http://www.me.berkeley.edu/gri_mech/

[Tee] L.S. Tee, S. Gotoh, and W.E. Stewart; "Molecular Parameters for Normal Fluids: The Lennard-Jones 12-6 Potential"; Ind. Eng. Chem., Fundamentals (1966), 5(3), 346-63 Table III: Correlation iii

[Lee] B.I. Lee and M.G. Kesler; "A Generalized Thermodynamic Correlation Based on Three-Parameter Corresponding States"; AIChE J., (1975), 21(3), 510-527

[Reid] R.C. Reid, J.M. Prausnitz and B.E. Poling; "The Properties of Gases and Liquids" 4th ed. McGraw-Hill, New York, NY, 1987

[Joback] K.G. Joback; Master's thesis, Chemical Engineering Department, Massachusetts Institute of Technology, Cambridge, MA, 1984

[Chang2000] A.Y. Chang, J.W. Bozzelli, and A. M. Dean. "Kinetic Analysis of Complex Chemical Activation and Unimolecular Dissociation Reactions using QRRK Theory and the Modified Strong Collision Approximation." *Z. Phys. Chem.* **214** (11), p. 1533-1568 (2000).

[Green2007] N.J.B. Green and Z.A. Bhatti. "Steady-State Master Equation Methods." *Phys. Chem. Chem. Phys.* **9**, p. 4275-4290 (2007).

[Susnow1997] R. G. Susnow, A. M. Dean, W. H. Green, P. K. Peczak, and L. J. Broadbelt. "Rate-Based Construction of Kinetic Models for Complex Systems." *J. Phys. Chem. A* **101**, p. 3731-3740 (1997).

[Gilbert1990] R. G. Gilbert and S. C. Smith. *Theory of Unimolecular and Recombination Reactions.* First Edition. Oxford, England: Blackwell Scientific (1990).

[Holbrook1996] K. A. Holbrook, M. J. Pilling, and S. H. Robertson. *Unimolecular Reactions.* Second Edition. Chichester, England: John Wiley and Sons (1996).

[Forst2003] W. Forst. *Unimolecular Reactions: A Concise Introduction.* First Edition. Cambridge, England: Cambridge University Press (2003).

[VariFlex] S. J. Klippenstein, A. F. Wagner, R. C. Dunbar, D. M. Wardlaw, and J. A. Miller. *VariFlex*. Sandia National Laboratory (2002).

[MultiWell] J. R. Barker, N. F. Ortiz, J. M. Preses, L. L. Lohr, A. Maranzana, and P. J. Stimac. "MultiWell-2008.3 Software." University of Michigan. http://aoss.engin.umich.edu/multiwell/ (2002).

[Barker2001] J. R. Barker. "Multiple-well, multiple-path unimolecular reaction systems. I. MultiWell computer program suite." *Int. J. Chem. Kin.* **33** (4), p. 232-245 (2001).

[CHEMDIS] A. Y. Chang and J. W. Bozzelli and A. M. Dean. "Kinetic analysis of complex chemical activation and unimolecular dissociation reactions using QRRK theory and the modified strong collision approximation." *Z. Phys. Chem.* **214** (11), p. 1533-1568 (2000).

[Knyazev1994] V. D. Knyazev, I. A. Dubinsky, I. R. Slagle, and D. Gutman. "Unimolecular Decomposition of t-C4H9 Radical." *J. Phys. Chem.* **98** (20), p. 5279-5289 (1994).

[Benson1976] S. W. Benson. *Thermochemical Kinetics: Methods for the Estimation of Thermochemical Data and Rate Parameters.* First Edition. New York, NY: John Wiley and Sons (1976).

[McQuarrie1973] D. A. McQuarrie. *Statistical Thermodynamics.* First Edition. New York, NY: Harper & Row (1973).

[Hill1986] T. L. Hill. *An Introduction to Statistical Thermodynamics.* Second Edition. New York, NY: Dover (1986).

[DQED] R. Hanson and F. Krogh. *DQED*. Sandia National Laboratory. http://www.netlib.org/opt/dqed.f.

[DQED90] J. Burkardt. *DQED FORTRAN90*. Florida State University. http://people.sc.fsu.edu/~burkardt/f_src/dqed/dqed.html.

[Astholz1979] D. C. Astholz, J. Troe, and W. Wieters. "Unimolecular processes in vibrationally highly excited cycloheptatrienes. I. Thermal isomerization in shock waves." *J. Phys. Chem.* **70** (11), p. 5107-5166 (1979).

[Beyer1973] T. Beyer and D.F. Swinehart. "Number of multiply-restricted partitions." *Comm. ACM* **16** (6), p. 379-379 (1973).

[JingThesis] Jing Song. PhD thesis, Building Robust Chemical Reaction Mechanisms: Next Generation Automatic Model Construction Software. (MIT, 2004)

[GreenChapter] W. H. Green, "Predictive Kinetics: A New Approach for the 21st Century" Advances in Chemical Engineering, vol. 32.

INDEX