

***MSTor:***  
**A program for calculating partition functions, free energies,  
enthalpies, entropies, and heat capacities of complex molecules  
including torsional anharmonicity**

**Manual**

Jingjing Zheng, Steven L. Mielke, Junwei Lucas Bao, Rubén Meana-Pañeda  
*Department of Chemistry, Chemical Theory Center, and Minnesota Supercomputing  
Institute, University of Minnesota, Minneapolis, MN 55455*

Kenneth L. Clarkson  
*IBM Almaden Research Center, San Jose, CA 95120*

and Donald G. Truhlar  
*Department of Chemistry, Chemical Theory Center, and Minnesota Supercomputing  
Institute, University of Minnesota, Minneapolis, MN 55455*

Program version: 2017-B

Program version date: Nov. 20, 2017

Manual updated date: Nov. 20, 2017

Copyright 2012, 2013, 2017

**Abstract:** *MSTor* is a computer program for calculating gas-phase molecular partition functions and thermodynamic functions (standard state energy, enthalpy, entropy, free energy, and heat capacity at constant pressure) as functions of temperature by the multi-structural approximation with torsional anharmonicity (MS-T). The MS-T approximation accounts for the coupling of torsions to one another and to overall rotation and, to some extent, the coupling between torsions and other vibrational modes. The program also calculates partition functions and thermodynamic functions by the multi-structural local harmonic (MS-LH) approximation or multi-structural local quasiharmonic (MS-LQ) approximation. This program package also includes seven utility codes that can be used as stand-alone programs. One utility program calculates reduced moments of inertia by the method of Kilpatrick and Pitzer, one generates conformational structures, the third and fourth calculate volumes of torsional subdomains defined by Voronoi tessellation either analytically or by Monte Carlo sampling, the fifth and the sixth generates template input files, and the seventh calculates one-dimensional torsional partition functions using the torsional eigenvalue summation method. One utility code for computing dual level MS-T partition functions is also available.

## Table of contents

Manual .....	1
1. Introduction .....	4
2. Licensing.....	5
3. Citations for the MS-T methods and the <i>MSTor</i> program.....	6
3.1. References for incorporated code .....	6
4. Distribution and installation.....	7
4. 1. Distribution.....	7
4. 2. Installation.....	8
4. 3. Description of Executables .....	9
5. Theoretical background.....	11
5. 1. Notation for the MS methods.....	12
5. 2. Translational partition function.....	13
5. 3. Electronic partition function .....	13
5. 4. Conformational–rovibrational partition function.....	13
5. 5. Treatment of rotational symmetry and mirror images .....	17
5. 6. Voronoi tessellation scheme for <i>M</i> values .....	18
5. 7. Low-temperature limits.....	20
5. 8. Transition structures .....	21
5. 9. Thermodynamics convention.....	21
6. Guidelines to generate the <i>MSTor</i> input file with <i>Gaussian</i> output files .....	22
7. Input files.....	24
7. 1. Input files for <i>mstor.exe</i> executable .....	24
7. 1. A. Description of sections of the main input file .....	25
7. 1. B. Glossary of keywords in \$GENERAL section .....	25
7. 1. C. Description of \$FRAMECHAIN section.....	28
7. 1. D. Description of \$FRAMEDEF section .....	29
7. 1. E. Description of \$INTDEF section.....	30
7. 1. F. Description of \$TEMP section .....	31
7. 1. G. Description of \$STRUCTURE section .....	31
7. 1. H. Description of the <i>hess.dat</i> input file .....	34
7. 2. Input files for the <i>ConfGen.exe</i> executable.....	35
7. 3. Input files for the <i>kpmoments.exe</i> executable.....	36
7. 4. Input files for the <i>mcvorm.exe</i> executable.....	37
7. 5. Input files for the <i>msinput.exe</i> executable.....	37
7. 6. Running the <i>mvinput.exe</i> executable .....	38
7. 7. Input files for the <i>tes.exe</i> executable.....	39
7. 8. Input files for the <i>vorm.exe</i> executable.....	40
7. 9 Input files for <i>DLMSTor.exe</i> executable .....	41
8. Test suites .....	42
8. 1. Test runs for <i>ConfGen.exe</i> .....	42
8. 2. Test runs for <i>kpmoments.exe</i> .....	43
8. 3. Test runs for <i>mcvorm.exe</i> .....	43
8. 4. Test runs for <i>msinput.exe</i> .....	43
8. 5. Test runs for <i>mvinput.exe</i> .....	44

8. 6. Test runs for <i>mstor.exe</i> .....	44
8. 7. Test runs for <i>tes.exe</i> .....	44
8. 8. Test runs for <i>vorm.exe</i> .....	45
9. Computers, operating systems, and compilers on which the code has been tested .....	46
10. Acknowledgments .....	49
11. Revision history .....	50

## 1. Introduction

The *MSTor* program is a computer program for calculating partition functions and thermodynamic functions of molecules by the multi-structural approximation with torsional anharmonicity (MS-T). There are two main variants of the algorithm:

- MS-T(U), which is an "uncoupled" approximation with respect to potential coupling between the torsional degrees of freedom
- MS-T(C) scheme, which is a "coupled" approximation in that it includes an approximate treatment of the potential coupling between torsions.

Both schemes approximately account for kinetic coupling between the torsions, for the coupling between the torsions and the remaining vibrational modes, and for coupling of the vibrations to overall rotation. The MS-T methods are designed to yield the harmonic-oscillator (HO) results in the low-temperature limit (the U scheme exactly reproduces this limit, whereas the C scheme retains a residual correction to the HO limit, which may be regarded as an approximate anharmonicity correction) and to yield free-rotor results for torsions in the high-temperature limit. The harmonic results are adjusted by torsional correction factors that are determined using either an uncoupled torsional potential or a coupled torsional potential in order to obtain a more accurate approximation in the intermediate-temperature regime. The MS-T method does not require assigning torsions to specific normal modes. Therefore the *MSTor* program can handle cases in which the system has significant coupling between torsions or between torsions and other modes, even at the normal mode level.

This program package also includes utilities that can help the user to generate the input files for the code or to generate comparison results. These additional codes are explained in more detail in Section 7. Most of the codes are designed for working with *Gaussian* outputs and fchk files; the utility codes that read from their corresponding input files (including mcvorm.exe, vorm.exe, tes.exe, kpmoments.exe and symmetry.exe) do not require any explicit operations on *Gaussian* files (although the automatic generation of the input files for mcvorm.exe and vorm.exe do require such operations).

## 2. Licensing

*MSTor* is copyrighted. Usage of *MSTor* requires a license, which may be obtained at

<http://comp.chem.umn.edu/mstor>.

or from the CPC Program Library. Usage of the code implies acceptance of the license provisions, which are:

1. No user or site will redistribute the source code or executable code to a third party in original or modified form without written permission of the principal investigator (Donald G. Truhlar) except as explicitly permitted by the following paragraph (numbered item 2). The licensee has no ownership rights in the software or in any copyrights for the software or documentation through this license.
2. A user/group license entitles the licensee (a person) and his/her research group and collaborators to use the program and share the source or executable code for use within a single research group. A site license enables the licensee (company, organization, or computing center) to install the program and allow access to the executable code to any number of users at that company, organization, or computing center, respectively.
3. Publications resulting from using the software will reference the program. References for the methods and the program are given in this manual.
4. No guarantee is made that the program is bug-free or suitable for specific applications, and no liability is accepted for any limitations in the mathematical methods and algorithms used within the program.
5. No consulting or maintenance services are guaranteed or implied.

### 3. Citations for the MS-T methods and the *MSTor* program

#### MS-T(U) method (based on an uncoupled torsional potential)

1. “Practical Methods for Including Torsional Anharmonicity in Thermochemical Calculations on Complex Molecules: The Internal-Coordinate Multi-Structural Approximation,” J. Zheng, T. Yu, E. Papajak, I. M. Alecu, S. L. Mielke, and D. G. Truhlar, *Physical Chemistry Chemical Physics* **13**, 10885–10907 (2011).  
doi.org/10.1039/C0CP02644A

#### MS-T(C) method (based on a coupled torsional potential)

2. “Quantum Thermochemistry: Multi-Structural Method with Torsional Anharmonicity Based on a Coupled Torsional Potential,” J. Zheng and D. G. Truhlar, *Journal of Chemical Theory and Computation* **9**, 1356-1367 (2013).  
doi.org/10.1021/ct3010722

#### Dual level MS-T method (which works for both coupled and uncoupled methods)

3. “Dual-Level Method for Estimating Multi-Structural Partition Functions with Torsional Anharmonicity” J. L. Bao, L. Xing, D. G. Truhlar, *Journal of Chemical Theory and Computation*. **13**, 2511-2522 (2017). doi.org/10.1021/acs.jctc.7b00232

#### *MSTor* program

4. J. Zheng, S. L. Mielke, J. L. Bao, R. Meana-Pañeda, K. L. Clarkson and D. G. Truhlar, *MSTor* computer program, version 2017-B, University of Minnesota, Minneapolis, MN, 2017.
5. “*MSTor*: A program for calculating partition functions, free energies, enthalpies, entropies, and heat capacities of complex molecules including torsional anharmonicity,” J. Zheng, S. L. Mielke, K. L. Clarkson, and D. G. Truhlar, *Computer Physics Communications* **183**, 1803-1812 (2012). doi.org/10.1016/j.cpc.2012.03.007
6. “*MSTor* version 2013: A new version of the computer code for the multistructural torsional anharmonicity with a coupled torsional potential”, J. Zheng, R. Meana-Pañeda, and D. G. Truhlar, *Computer Physics Communications* **184**, 2032-2033 (2013).  
doi.org/10.1016/j.cpc.2013.03.011

### 3.1. References for incorporated code

1. The code for evaluating of the point group of symmetry of a structure is from Serguei Patchkovskii (<http://www.cobalt.chem.ucalgary.ca/ps/symmetry>) and is redistributed under GNU general public license in the *MSTor* package.

## 4. Distribution and installation

### 4.1. Distribution

The distributed *MSTor* package, *mstor.tar.gz*, contains the following directories and files:

*exe*: This is an empty directory. After successful installation, this directory should contain eleven executable files as described in Section 4. C.

*hull*: This has a subdirectory *src/* that contains the source code of the *hull* program.

*install.pl*: This is a Perl script that can be used for installation of the *MSTor* program.

*src*: This directory contains source code for the main program *mstor.exe*

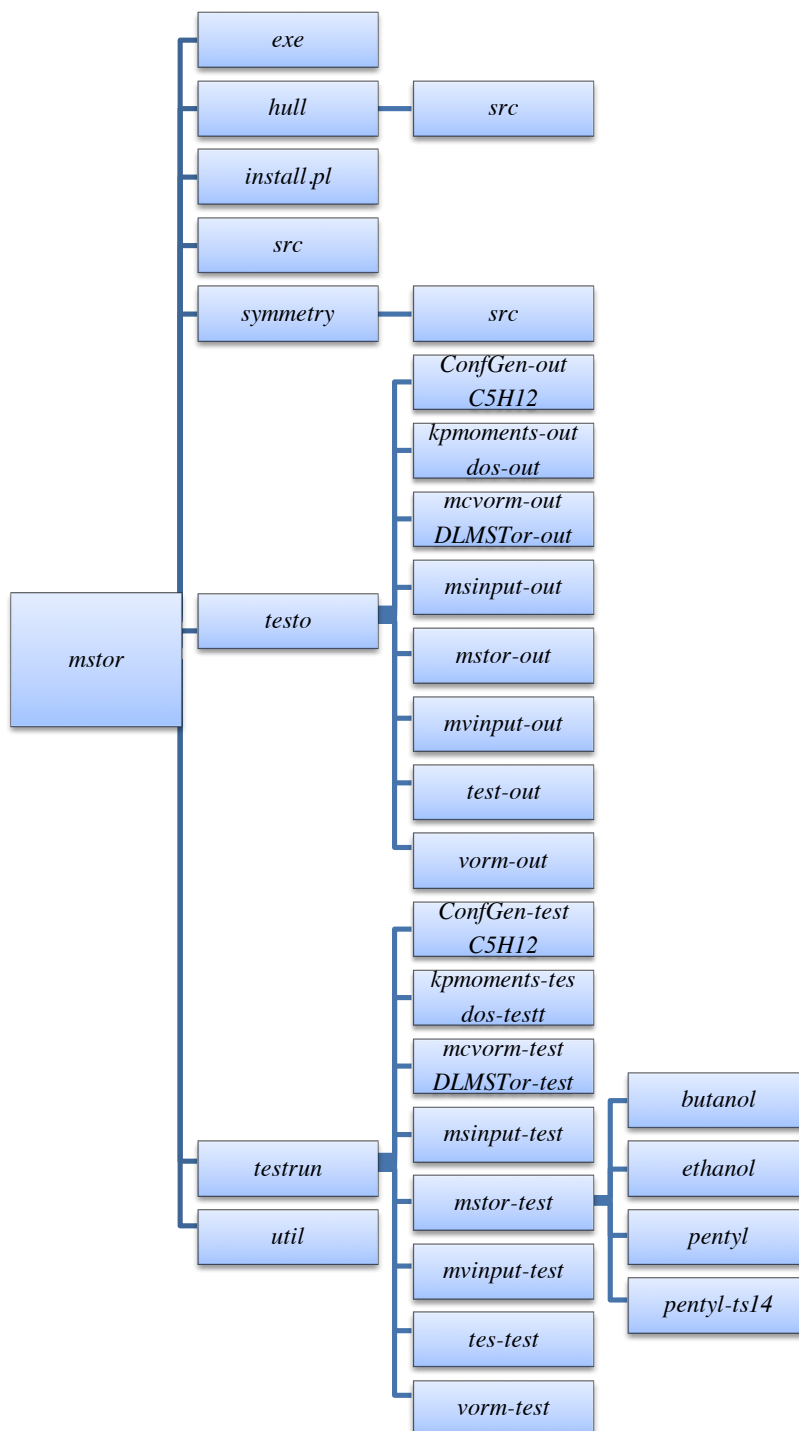
*symmetry*: This has a subdirectory *src/* that contains the source code of the *symmetry* program.

*testo*: This directory contains eight subdirectories, *ConfGen-out*, *kpmoments-out*, *mcvorm-out*, *msinput-out*, *mvinput-out*, *mstor-out*, *tes-out*, and *vorm-out*. The eight subdirectories contain output files of test runs for the *mstor.exe* program and the seven utility programs, respectively.

*testrun*: This directory contains eleven subdirectories: *C5H12*, *ConfGen-test*, *DLMSTor-test*, *dos-test*, *kpmoments-test*, *mcvorm-test*, *msinput-test*, *mvinput-test*, *mstor-test*, *tes-test*, and *vorm-test*. The eight subdirectories contain input files of test runs for the *mstor.exe* program and the seven utility programs, respectively. The *mstor-test* directory also has four subdirectories that contain test files for different molecules including one test at a saddle point.

*util*: This directory contains the source code of eight utility programs.

The directory structure is shown in the following figure.



## 4. 2. Installation

The installation procedure consists of the following steps:



- 1) Copy the *mstor.tar.gz* package to the your home directory and then unzip and untar the package.  
`tar -xzf <Return>`
- 2) Go to the *mstor/* directory and run the installation script  
`./install.pl <Return>`  
 The *install.pl* script will search for available Fortran and C compilers and compile the *MSTor* code in the *mstor/src/* directory. It will also compile the *hull* code in the *mstor/hull/* directory, and it will compile all utility codes in the *mstor/utli/* directory.
- 3) If the install script runs successfully, the *mstor/exe/* directory will contain ten executables: *ConfGen.exe*, *kpmoments.exe*, *mcvorm.exe*, *mstor.exe*, *hull.exe*, *msinput.exe*, *mvinput.exe*, *symmetry.exe*, *tes.exe*, and *vorm.exe*.
- 4) Add the *mstor/exe/* directory to your Linux/Unix PATH variable (source `~/bashrc`).

### 4.3. Description of Executables

Executable	Description	Usage
<i>mstor.exe</i>	Main executable for calculating the MS-T partition function, the MS-LH partition function, and thermodynamic functions	<i>mstor.exe</i> < mstor.inp > mstor.out
<i>vorm.exe</i>	Utility code to calculate local periodicities ( <i>M</i> values) using Voronoi tessellation by calling <i>hull.exe</i>	<i>vorm.exe</i> < mvorm.inp > mvorm.out
<i>mcvorm.exe</i>	Utility code to calculate local periodicities ( <i>M</i> values) using the Monte Carlo method	<i>mcvorm.exe</i> < mvorm.inp > mvorm.out
<i>msinput.exe</i>	Utility code to generate a template input file for <i>mstor.exe</i> . If the <i>mvorm.out</i> file exists, the code also reads the local periodicities from the <i>mvorm.out</i> file.	<i>msinput.exe</i> < all.fchk
<i>mvinput.exe</i>	Utility code to generate a template input file for <i>vorm.exe</i> and/or <i>mcvorm.exe</i> , and to pick out distinguishable structures (exclude mirror images)	<i>./mvinput.exe</i>
<i>hull.exe</i>	Computes the convex hull of a point set in general dimension. It is used by <i>vorm.exe</i> to calculate each Voronoi cell volume	see <a href="http://www.netlib.org/voronoi/hull.html">http://www.netlib.org/voronoi/hull.html</a> for more details about using this as a standalone program.
<i>symmetry.exe</i>	Determines the symmetry point group of the molecular structure.	see <a href="http://www.cobalt.chem.ucalgary.ca/ps/symmetry">http://www.cobalt.chem.ucalgary.ca/ps/symmetry</a> for more details, and to use it as a standalone program.

<i>ConfGen.exe</i>	Utility code to generate a set of conformational structures by rotating user specified bonds in an input structure	<i>ConfGen.exe</i> < input
<i>kpmoments.exe</i>	Utility code to compute reduced moments of inertia by the method of Kilpatrick and Pitzer	<i>kpmoments.exe</i> < input > output
<i>tes.exe</i>	Utility code to calculate 1-D torsional partition functions using the torsional eigenvalue summation method	<i>tes.exe</i> < input > output
<i>DLMSTor.exe</i>	Utility code to create input mstor.inp for dual level MS-T calculations.	<i>/DLMSTor.exe</i>

## 5. Theoretical background

The total partition function is calculated here by

$$Q_{\text{total}} = Q_{\text{trans}} Q_{\text{elec}} Q_{\text{con-rovib}} \quad (1)$$

where  $Q_{\text{trans}}$  is the translational partition function,  $Q_{\text{elec}}$  is the electronic partition function, and  $Q_{\text{con-rovib}}$  is the conformational–rovibrational partition function. Equation (1) assumes that the translational partition function and electronic degrees of freedom are separable from the other degrees of freedom. In all equations, the zero of energy for conformational–rovibrational partition functions is at the local minimum of the potential energy function, not at the lowest-energy vibrational state. Notice that this differs from the usual textbook convention (see section 5.8).

The thermodynamic functions for the Gibbs free energy ( $G^\circ$ ), average energy ( $E^\circ$ ), enthalpy ( $H^\circ$ ), and entropy ( $S^\circ$ ) are calculated analytically as

$$G^\circ = -\ln(Q)/\beta + k_B T \quad (2)$$

$$E^\circ = -\frac{\partial \ln(Q)}{\partial \beta} \quad (3)$$

$$H^\circ = E^\circ + P^\circ V \quad (4)$$

$$S^\circ = k_B \ln Q - \frac{1}{T} \left( \frac{\partial \ln Q}{\partial \beta} \right) \quad (5)$$

where  $\beta = 1/k_B T$ ,  $k_B$  is the Boltzmann's constant,  $T$  is temperature,  $P$  is pressure, and  $V$  is volume. The degree symbol ( $^\circ$ ) denotes standard state. The default for the standard state pressure is 1 bar, which equals 100 kPa or 0.987 atm.

Heat capacity at constant pressure ( $C_p^\circ$ ) is calculated using a four-point central finite differences formula

$$C_p^\circ(T) = \frac{H^\circ(T - 2\delta T) - 8H^\circ(T - \delta T) + 8H^\circ(T + \delta T) - H^\circ(T + 2\delta T)}{12\delta T} \quad (6)$$

where  $\delta T$  is a step size for temperature.

### 5.1. Notation for the MS methods

The program calculates two MS approximations: MS-T (multistructural method for torsional anharmonicity) and MS-LH (multistructural local harmonic approximation). These methods were denoted in the original publication by other names as indicated in the chart below. The new names are recommended for future usage.

New name	Original or other name		
MS-T	MS-AS <sup>a</sup>	MS-AS-T <sup>b</sup>	
MS-LH	MS-HO <sup>a</sup>	MS-AS-HO <sup>b</sup>	MS-LQ <sup>c</sup>

<sup>a</sup> J. Zheng, T. Yu, E. Papajak, I. M. Alecu, S. L. Mielke, D. G. Truhlar, Phys. Chem. Chem. Phys. 13 (2011), 10885.

<sup>b</sup> T. Yu, J. Zheng, D. G. Truhlar, Chem. Sci. 2 (2011), 2199.

<sup>c</sup> When one uses scaled frequencies<sup>1</sup> in the MS-LH method, it is actually locally quasiharmonic (LQ or QH) because the scaling accounts not only for systematic errors in the electronic structure method, but also for anharmonicity of the high-frequency modes. In such a case the method may still be called MS-LH, because the formulas are still based on the harmonic oscillator, or it may be called MS-LQ<sup>2</sup>, where LQ denotes local quasiharmonic.

The MS-T method can either be based on an uncoupled torsional potential or be based on a coupled torsional potential. For both the coupled and uncoupled MS-T method, there is also a dual-level version.<sup>3</sup> To distinguish the methods, we use specific names as follows:

Name	Method	Reference
MS-T(U)	MS-T with uncoupled torsional potential	1
MS-T(C)	MS-T with coupled torsional potential (default)	2
Dual level MS-T	dual-level MS-T (works for both C and U versions)	3

<sup>1</sup> I. M. Alecu, J. Zheng, Y. Zhao, and D. G. Truhlar, J. Chem. Theory Comp. 6, 2872-2887 (2010).

<sup>2</sup> E. Papajak, P. Seal, X. Xu, and D. G. Truhlar, J. Chem. Phys. 137, 064110/1-8 (2012).

<sup>3</sup> J. L. Bao, L. Xing, D. G. Truhlar, Journal of Chemical Theory and Computation. 13, 2511-2522 (2017)

## 5.2. Translational partition function

The translational partition function for an ideal molecular gas is given by

$$Q_{\text{trans}} = \left( \frac{mk_B T}{2\pi\hbar^2} \right)^{3/2} V^\circ \quad (7)$$

where  $V^\circ$  is the volume per particle in the standard state,  $k_B$  is the Boltzmann's constant,  $T$  is temperature,  $\hbar$  is Planck's constant divided by  $2\pi$ , and  $m$  is the molecule's mass. For an ideal gas,  $V^\circ = k_B T / P^\circ$ , where  $P^\circ$  is the pressure in the standard state.

Often we quote the answer as a translational partition function per unit volume

$$\Phi_{\text{trans}} = Q_{\text{trans}} / V^\circ \quad (8)$$

Similarly, the total partition function per unit volume is

$$\Phi_{\text{total}} = Q_{\text{total}} / V^\circ = \Phi_{\text{total}} Q_{\text{elec}} Q_{\text{con-rovib}} \quad (9)$$

## 5.3. Electronic partition function

The electronic partition function is calculated by

$$Q_{\text{elec}} = \sum_i d_i e^{-\varepsilon_i / k_B T} \quad (10)$$

where  $d_i$  and  $\varepsilon_i$  are the degeneracy and energy of the electronic state  $i$ , respectively. Here we set the ground electronic state energy as zero ( $\varepsilon_1 = 0$ ). The partial derivative of  $Q_{\text{elec}}$  with respect to  $\beta$  is

$$-\frac{\partial \ln Q_{\text{elec}}}{\partial \beta} = \frac{1}{Q_{\text{elec}}} \sum_i d_i \varepsilon_i e^{-\varepsilon_i \beta} \quad (11)$$

## 5.4. Conformational–rovibrational partition function

### 5.4.1 MS-T(C) method

When torsions are coupled in the MS-T(C) scheme, we assume that each coupled torsion  $\eta$  for a structure  $j$  has the reference potential form, that is,

$$V_{j,\eta} = U_j + \frac{W_{j,\eta}^{(C)}}{2} \left[ 1 - \cos M_{j,\eta} (\phi_{j,\eta} - \phi_{j,\eta,\text{eq}}) \right]; \quad \frac{-\pi}{M_{j,\eta}} \leq \phi_{j,\eta} - \phi_{j,\eta,\text{eq}} \leq \frac{\pi}{M_{j,\eta}} \quad (12)$$

where  $U_j$  is the energy of structure  $j$  (where the global minimum is set to 0),  $W_{j,\eta}^{(C)}$  is effective coupled barrier height,  $M_{j,\eta}$  is a local periodicity parameter,  $\phi_{j,\eta}$  is torsional coordinate, and  $\phi_{j,\eta,\text{eq}}$  is the torsional coordinate at equilibrium geometry.

For a molecule or a transition structure that has  $J$  distinguishable structures and  $t$  torsions, the conformational–rovibrational partition function according to the MS-T(C) method is

$$Q_{\text{con-rovib}}^{\text{MS-T(C)}} = \sum_{j=1}^J Q_{\text{rot},j} \exp(-\beta U_j) Q_j^{\text{HO}} \prod_{\eta=1}^t f_{j,\eta} \quad (13)$$

$$Q_j^{\text{HO}} = \prod_{i=1}^F \frac{\exp(-\beta \hbar \omega_{j,i}/2)}{1 - \exp(-\beta \hbar \omega_{j,i})} \quad (14)$$

where  $Q_{\text{rot},j}$  is the rotational partition function of structure  $j$ ,  $\beta$  is  $1/k_B T$ ,  $k_B$  is Boltzmann's constant,  $T$  is temperature,  $Q_j^{\text{HO}}$  is the usual normal-mode harmonic oscillator vibrational partition function calculated at structure  $j$ ,  $f_{j,\eta}$  is a factor that takes account of torsional anharmonicity,  $F$  is the number of degrees of freedom for the vibrational modes, and  $\omega_{j,i}$  denotes the normal-mode vibrational frequency of mode  $i$  of structure  $j$ . Note that the zero of energy for  $Q_j^{\text{HO}}$  is at the local minimum of the potential energy function for structure  $j$ , not at the zero point level of structure  $j$ .

We use the classical expression for the rotational partition function for structure  $j$

$$Q_{\text{rot},j} = \frac{\sqrt{\pi}}{\sigma_{\text{rot},j}} \left( \frac{2}{\hbar^2 \beta} \right)^{3/2} \sqrt{I_{A,j} I_{B,j} I_{C,j}} \quad (15)$$

where  $\sigma_{\text{rot},j}$  is the symmetry number of overall rotation, and  $I_{A,j}$ ,  $I_{B,j}$ , and  $I_{C,j}$  are the principal moments of inertia.

If the  $f_{j,\eta}$  are set to unity, the partition function  $Q_{\text{con-rovib}}^{\text{MS-T(C)}}$  reduces to the multi-structural local-harmonic (MS-LH) partition function. Note that the resulting method is called "local harmonic" but the result is actually quasiharmonic if one uses frequencies scaled<sup>1</sup> to account for anharmonicity (and possibly also accounting for other factors).

Because the torsional anharmonicity correction is based on the coupled torsional potential, the individual  $f_{j,\eta}$  is not meaningful and the product of  $f_{j,\eta}$  is the correction for torsional anharmonicity for the  $t$  coupled torsions,

$$\prod_{\eta=1}^t f_{j,\eta} = (2\pi\beta)^{t/2} \frac{\prod_{m=1}^F \omega_{j,m}}{\prod_{\bar{m}=1}^{\bar{F}} \bar{\omega}_{j,\bar{m}}} \frac{\sqrt{\det \mathbf{D}_j}}{\prod_{\tau=1}^t M_{j,\tau}} \prod_{\eta=1}^t \exp(-\beta W_{j,\eta}^{(C)}/2) I_0(\beta W_{j,\eta}^{(C)}/2) \quad (16)$$

where  $\bar{\omega}_{j,\bar{m}}$  are torsion-projected normal mode frequencies,  $\mathbf{D}_j$  are the Kilpatrick and Pitzer torsional moment of inertia matrices,  $M_{j,\tau}$  are local periodicity parameters for uncoupled torsion  $\tau$ , and  $I_0$  is a modified Bessel function. Note that it is difficult to determine the coupled periodicity parameter  $M_{j,\eta}$  so that we use the  $M_{j,\tau}$  here instead of  $M_{j,\eta}$ .

To calculate the thermodynamic functions from the MS-T(C) partition function, we need the partial derivative of the logarithm of the partition function with respect to  $\beta$ :

$$\begin{aligned}
-\frac{\partial}{\partial \beta} \ln Q_{\text{con-rovib}}^{\text{MS-T(C)}} &= \frac{1}{Q_{\text{con-rovib}}^{\text{MS-T(C)}}} \sum_{j=1}^J \left( Q_{\text{rot},j} \exp(-\beta U_j) Q_j^{\text{HO}} \prod_{\eta=1}^t f_{j,\eta} \right) \left\{ \frac{3}{2\beta} + U_j \right. \\
&+ \sum_{m=1}^F \frac{\hbar \omega_{j,m}}{2} \left( \frac{1 + e^{-\beta \hbar \omega_{j,m}}}{1 - e^{-\beta \hbar \omega_{j,m}}} \right) + \sum_{\eta=1}^t \left[ \frac{W_{j,\eta}^{(\text{C})}}{2} \left( 1 - \frac{I_1(\beta W_{j,\eta}^{(\text{C})}/2)}{I_0(\beta W_{j,\eta}^{(\text{C})}/2)} \right) - \frac{1}{2\beta} \right] \left. \right\} \quad (17)
\end{aligned}$$

#### 5.4.2 MS-T(U) method

In the MS-T(U) method, we assume each uncoupled torsion  $\tau$  has the potential as

$$V_{j,\tau} = U_j + \frac{W_{j,\tau}^{(\text{U})}}{2} \left[ 1 - \cos M_{j,\tau} (\phi_{j,\tau} - \phi_{j,\tau,\text{eq}}) \right]; \quad \frac{-\pi}{M_{j,\tau}} \leq \phi_{j,\tau} - \phi_{j,\tau,\text{eq}} \leq \frac{\pi}{M_{j,\tau}} \quad (18)$$

where  $W_{j,\tau}^{(\text{U})}$  is effective uncoupled barrier height,  $\phi_{j,\tau}$  is torsional coordinate, and  $\phi_{j,\tau,\text{eq}}$  is the torsional coordinate at equilibrium geometry

For a molecule that has  $J$  distinguishable structures and  $t$  torsions, the conformational-rovibrational partition function according to the MS-T(U) method is

$$Q_{\text{con-rovib}}^{\text{MS-T(U)}} = \sum_{j=1}^J Q_{\text{rot},j} \exp(-\beta U_j) Q_j^{\text{HO}} Z_j \prod_{\tau=1}^t f_{j,\tau} \quad (19)$$

where  $Z_j$  is a factor designed to ensure that the MS-T(U) partition function reaches the correct high-temperature limit, and  $f_{j,\tau}$  is an internal-coordinate torsional anharmonicity function that, in conjunction with  $Z_j$ , adjusts the harmonic result of structure  $j$  for the presence of the torsional motion  $\tau$ .

If the  $Z_j$  and  $f_{j,\tau}$  are set to unity, the partition function  $Q_{\text{con-rovib}}^{\text{MS-T}}$  reduces to the multi-structural local-harmonic (MS-LH) partition function.

The torsional anharmonicity function using uncoupled internal-coordinate approximation for torsion  $\tau$  at structure  $j$  is given

$$f_{j,\tau} = \frac{\bar{\omega}_{j,\tau} \sqrt{2\pi\beta I_{j,\tau}}}{M_{j,\tau}} \exp(-\beta W_{j,\tau}^{(\text{U})}/2) I_0(\beta W_{j,\tau}^{(\text{U})}/2) \quad (20)$$

where  $\bar{\omega}_{j,\tau}$  is an internal-coordinate torsional frequency,  $I_{j,\tau}$  is the torsional moment of inertia calculated by Pitzer's method without coupling between torsions,  $M_{j,\tau}$  is a local periodicity parameter, and  $W_{j,\tau}^{(\text{U})}$  is an uncoupled effective barrier height. These parameters are interrelated as

$$W_{j,\tau}^{(\text{U})} = \frac{2I_{j,\tau} \bar{\omega}_{j,\tau}^2}{M_{j,\tau}^2} \quad (21)$$

The factor  $Z_j$  is given by

$$Z_j = g_j + (1 - g_j) Z_j^{\text{int}} Z_j^{\text{coup}} \quad (22)$$

where  $Z_j^{\text{int}}$  replaces the normal-mode vibrational partition function in the high- $T$  limit by internal-coordinate ones, and  $Z_j^{\text{coup}}$  replaces the uncoupled moments of inertia for individual torsions by values that account for their coupling.  $g_j \rightarrow 1$  at low  $T$  where the effects of rotational-vibrational coupling are minimal, and  $g_j \rightarrow 0$  at high  $T$ . The  $Z_j^{\text{int}}$  are given by

$$Z_j^{\text{int}} = \frac{\prod_{m=1}^{F-t} \bar{\omega}_{j,m}^{-1} \prod_{\tau=1}^t \bar{\omega}_{j,\tau}^{-1}}{\prod_{m=1}^F \omega_{j,m}^{-1}} \quad (23)$$

where  $F$  is the number of vibrational degrees of freedom,  $\omega_{j,m}$  is the frequency of normal mode  $m$ . The factors  $Z_j^{\text{coup}}$  are equal to

$$Z_j^{\text{coup}} = \left( \frac{|\det \mathbf{D}_j|}{\prod_{\tau=1}^t I_{j,\tau}} \right)^{1/2} \quad (24)$$

The expression used for  $g_j$  to enforce the correct limits is

$$g_j = \left( \prod_{\tau=1}^t \tanh \frac{\sqrt{2\pi k_{j,\tau} \beta}}{M_{j,\tau}} \right)^{1/t} \quad (25)$$

where  $k_{j,\tau}$  is the force constant for torsion  $\tau$  at structure  $j$  in internal coordinates.

The partial derivative of the MS-T partition function with respect to  $\beta$  is

$$\begin{aligned} -\frac{\partial}{\partial \beta} \ln(Q_{\text{con-rovib}}^{\text{MS-T(U)}}) &= \frac{1}{Q_{\text{con-rovib}}^{\text{MS-AS}}} \sum_{j=1}^J \left\{ e^{-\beta U_j} Q_{\text{rot},j} Q_j^{\text{HO}} Z_j \prod_{\tau=1}^t f_{j,\tau} \left( \frac{3}{2\beta} + U_j \right. \right. \\ &\quad + \sum_{m=1}^F \frac{\hbar \omega_{j,m}}{2} \frac{1 + e^{-\beta \hbar \omega_{j,m}}}{1 - e^{-\beta \hbar \omega_{j,m}}} \\ &\quad \left. \left. - \left( g \frac{(1 - Z_j^{\text{int}} Z_j^{\text{coup}})}{2t \sqrt{\beta} Z_j} \left[ \sum_{\tau=1}^t \frac{\bar{\omega}_{j,\tau} \sqrt{2\pi I_{j,\tau}}}{M_{j,\tau}} \frac{\text{sech}^2(\bar{\omega}_{j,\tau} \sqrt{2\pi I_{j,\tau} \beta} / M_{j,\tau})}{\tanh(\bar{\omega}_{j,\tau} \sqrt{2\pi I_{j,\tau} \beta} / M_{j,\tau})} \right] \right) \right. \right. \\ &\quad \left. \left. + \sum_{\tau=1}^t \left( \frac{I_{j,\tau} \bar{\omega}_{j,\tau}^2}{M_{j,\tau}^2} - \frac{1}{2\beta} - \frac{I_{j,\tau} \bar{\omega}_{j,\tau}^2}{M_{j,\tau}^2} \frac{I_1(\beta I_{j,\tau} \bar{\omega}_{j,\tau}^2 / M_{j,\tau}^2)}{I_0(\beta I_{j,\tau} \bar{\omega}_{j,\tau}^2 / M_{j,\tau}^2)} \right) \right) \right\} \quad (26) \end{aligned}$$

Note that the zero-point energy corresponding to the MS-T partition function is

$$E_0^{\text{MS-LH}} = \min_j \{ E_{j,0}^{\text{HO}} + U_j \} \quad (27)$$

where  $E_{j,0}^{\text{HO}}$  is the harmonic oscillator zero-point energy of structure  $j$  with the zero of energy at its local minimum, and which is given by



$$E_{j,0}^{\text{HO}} = \frac{\hbar}{2} \sum_{m=1}^F \omega_{j,m} \quad (28)$$

### 5.5. Treatment of rotational symmetry and mirror images

Special care is needed when considering rotational symmetry for systems that exhibit torsional motion. To calculate the classical partition function for such a system, either one could integrate  $\exp(-\beta V)$  over only the symmetry unique portion of configuration space or one could integrate  $\exp(-\beta V)$  over all of configuration space and divide the result by a *structure-independent* symmetry number,  $\sigma_{\text{tot}}$ , given by

$$\sigma_{\text{tot}} = \sigma_{\text{grot}} \prod_{\tau=1}^t \sigma_{\tau,\text{tor}} \quad (29)$$

where  $\sigma_{\text{grot}}$  is a generalized rotational symmetry number that accounts for the number of ways that overall rotation can map a specific structure either onto itself or onto another structure that may be converted to the original structure purely by torsional motion. However, it is convenient to consider only the  $J$  fully-distinguishable structures (i.e., structures that are distinguishable after considering both torsional and rotational symmetry) in the structure summation of equations such as Eq. 13 or Eq. 19. When doing so, we use structure-specific rotational symmetry numbers,  $\sigma_{\text{rot},j} \leq \sigma_{\text{grot}}$  because we are then using a hybrid approach whereby we are accounting for rotational symmetry partly via the use of symmetry numbers and partly by omitting certain regions of configuration space that we recognize as symmetrically equivalent to other regions.

For many systems, the observed structures occur as pairs of mirror images (sometimes a structure is its own mirror image); mirror images are distinguishable, but because they have equal contributions to the overall partition function it is convenient to perform explicit calculations for only one member of each pair and to weight this structure by a factor of 2 in the final results. The various utility codes included with the software package identify mirror images in order to make the calculation more efficient, for example by reducing the required number of harmonic frequency calculations.

It is important to note that the Voronoi tessellation codes that calculate  $M$  parameters for strongly coupled torsions do not explicitly consider rotational symmetry; thus, one needs to provide structure information to these routines for all structures that would be distinct *without* accounting for rotational symmetry (see section 5.6 below).

The case of pentane is a useful illustration of how we treat rotation and mirror images. Pentane has four torsions: C1-C2, C2-C3, C3-C4, and C4-C5. If we considered only the two non-methyl torsional degrees of freedom (i.e., C2-C3 and C3-C4, each of which has a torsional symmetry number equal to 1), we would find 11 structures before accounting for symmetry. A plot of these structures using the data reported in R. A. Scott and H. A. Scheraga, J. Chem. Phys., 44, 3054 (1966) is given in Figure 5.1.

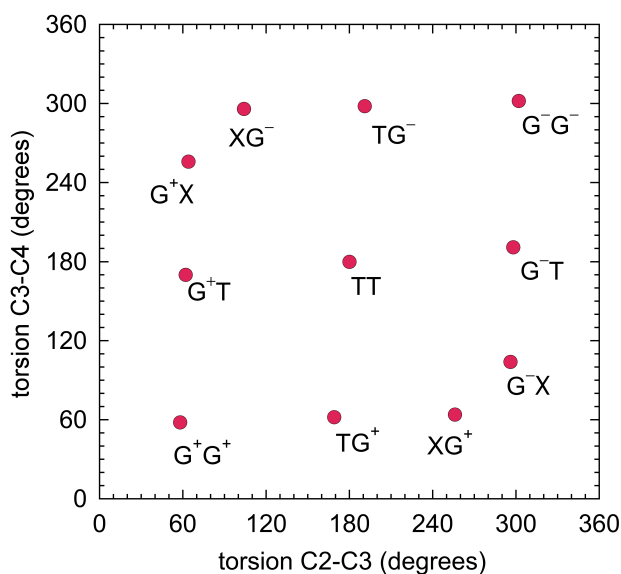


Figure 5.1. Structures for pentane.

We might have naively expected to find nine uniformly distributed structures in Fig. 5.1, but steric effects lead to finding two structures,  $G^+X$  and  $XG^-$ , nearby to  $(60, 300)$  where we might have expected to find one structure, and, similarly, two structures,  $XG^+$  and  $G^-X$ , nearby to where we might have expected a single structure at  $(300, 60)$ . For this system  $\sigma_{\text{grot}} = 2$ , but only three ( $TT$ ,  $G^+G^+$ , and  $G^-G^-$ ) of the 11 structures have  $\sigma_{\text{rot},j} = 2$ ; the eight remaining structures have  $\sigma_{\text{rot},j} = 1$ , but overall rotation maps each of these structures onto one of the other eight structures (for example,  $G^+T$  is related by an overall rotation to  $TG^+$ ), so only four of these eight are distinct after accounting for overall rotation. Of the seven distinct structures, one ( $TT$ ) is its own mirror image and the remaining six occur as three pairs of mirror images (for example,  $G^+T$  and  $G^-T$ ); thus, we need to perform frequency calculations for only four structures (for example,  $TT$ ,  $G^+G^+$ ,  $G^+T$ , and  $G^+X$ ) to obtain the final MS-T partition functions.

The correct way to handle the issues raised in this subsection is revisited in Section 6.

## 5.6. Voronoi tessellation scheme for $M$ values

When torsions are strongly coupled together, it is sometimes impossible to assign integer  $M_{j,\tau}$  values for each torsion. We divide the  $t$  torsions into two types: nearly separable (NS) and strong coupled (SC). We use the notation  $\text{NS}:\text{SC} = t_{\text{NS}} : t_{\text{SC}}$  to denote that  $t_{\text{NS}}$  torsions are treated as nearly separable and  $t_{\text{SC}}$  torsions are treated as strongly coupled. In general, the strongly coupled coordinates may be further partitioned into two or more subspaces, with each subspace involving only those coordinates that are strongly coupled to each other.

Each of the strongly coupled subspaces is treated by Voronoi tessellation separately. Voronoi tessellation divides a space into cells around a discrete set of points. In the applications considered here, the space to be tessellated is described by the dihedral angles  $\phi_1, \phi_2, \dots, \phi_{t_{\text{SC}}}$ , and the points correspond to structures. Each cell corresponds to a specific structure and consists of all torsional configurations closer to this structure than to any other structure when only the  $t_{\text{SC}}$  strongly coupled degrees of freedom are considered.

Next we define the concept of torsional symmetry number  $\sigma_\tau$  for torsion  $\tau$ ; if there are  $t$  torsions in a molecule, there will be  $t$  torsional symmetry numbers, each of which corresponds to one torsion in the molecule. The symmetry number for torsion  $\tau$  can be determined by treating the least symmetric of the two rotating fragments as a fixed frame and counting the number of identical structures obtained when the more symmetrical top is rotated from 0 to 360 degrees with respect to the torsional bond  $t$  and relaxed. For example, the symmetry numbers for C-O torsion in methanol, C-N torsion in nitromethane, and C-C torsion in 1,2 dichloroethane are 3, 3, and 1, respectively.

The tessellation code properly accounts for torsional symmetry according to the input torsional symmetry numbers  $\sigma_\tau$  in the MC sampling (see equation 30), but the current version does not exploit torsional symmetry efficiently to reduce the statistical uncertainty (which does not affect the computed  $M$  values). The code also does not use rotational symmetry to reduce the domain of configurations it must consider, so for systems exhibiting rotational symmetry (e.g., linear alkanes), structures that would be indistinguishable under rotational and torsional symmetry but are distinguishable under *only* torsional symmetry **must** be included in the input (the utility code `mvinput.exe` automatically includes the additional structures that are required provided that they were obtained in the prior structure search step). When the input contains pairs of structures that are mirror images of each other, and/or rotationally related to each other, the code will use this information to reduce the statistical uncertainties.

In this manual we will refer to the total number of fully-distinguishable structures (i.e., distinguishable after considering all possible symmetries) as  $J$ , and we will refer to the number of structures that would be distinguishable without considering rotational symmetry (but fully accounting for torsional symmetry) as  $\tilde{J}$ . (There may be places in previous writing where these two quantities were not carefully distinguished. For the case of pentane that we have discussed earlier and for which only the two nonmethyl torsions are included in the calculations, we would have  $J = 7$  and  $\tilde{J} = 11$ .) The volume of the SC torsional subspace is then given by

$$\Omega^{\text{SC,tot}} = \sum_{j=1}^{\tilde{J}} \Omega_j^{\text{SC}} = \frac{(2\pi)^{t_{\text{SC}}}}{\prod_{\tau=1}^{t_{\text{SC}}} \sigma_\tau} \quad (30)$$

where  $t_{\text{SC}}$  is the number of torsions in the strongly coupled space. Whenever we use Voronoi tessellation, we assume that the torsional subspace is so strongly coupled that we cannot assign  $M_{j,\tau_{\text{SC}}}$  by considering each torsion separately; then we replace all  $M_{j,\tau_{\text{SC}}}$  for strongly coupled torsions of a given  $j$  by a single  $M_j^{\text{SC}}$  equal to

$$M_j^{\text{SC}} = \frac{2\pi}{(\Omega_j^{\text{SC}})^{1/t_{\text{SC}}}} \quad (29)$$

## 5.7. Low-temperature limits

In this section, we use the common thermodynamics convention where temperature is written as a subscript. Then

$$G_T^\circ = -\ln(Q) / \beta + k_B T \quad (30)$$

$$E_T^\circ = -\frac{\partial \ln(Q)}{\partial \beta} \quad (31)$$

$$H_T^\circ = E_T^\circ + PV = E_T^\circ + RT = \varepsilon(T) + RT \quad (32)$$

$$S_T^\circ = k_B \ln Q - \frac{1}{T} \left( \frac{\partial \ln Q}{\partial \beta} \right) \quad (33)$$

Consider very low temperature ( $T \rightarrow 0$ ), where only the ground state, with energy  $\varepsilon^{\text{G}}$  and degeneracy  $d_0$ , is important. Note that if the structure that has the lowest zero-point-inclusive energy is the same as the structure that has the lowest zero-point-exclusive energy, then  $\varepsilon^{\text{G}}$  is the zero point vibrational energy of that structure. Otherwise  $\varepsilon^{\text{G}}$  is given by eq. (18). Then, where all arrows refer to the limit as temperature goes to zero, we have:

$$Q \rightarrow d_0 \exp(-\beta \varepsilon^{\text{G}}) \rightarrow 0 \quad (34)$$

Therefore

$$\ln Q \rightarrow -\beta \varepsilon^{\text{G}} + \ln d_0 \quad (35)$$

$$-\frac{\partial}{\partial \beta} \ln Q \rightarrow \varepsilon^{\text{G}} \quad (36)$$

Substituting eqs. (29)–(31) into eqs. (25)–(27) yields the following low- $T$  limits:

$$G_T^\circ \rightarrow \varepsilon^{\text{G}} \quad (37)$$

$$E_T^\circ \rightarrow \varepsilon^{\text{G}} \quad (38)$$

$$H_T^\circ \rightarrow \varepsilon^{\text{G}} \quad (39)$$

Note that most statistical mechanics textbooks use a different zero of energy, at the ground state level. We label partition functions with that choice as  $\tilde{Q}$  where

$$\tilde{Q} \equiv Q \exp(\beta \epsilon^G) \quad (40)$$

and we note that the limits of eqs. (30)–(32) would be different with that different choice of the zero of energy

Equation (28) yields

$$S_T^\circ \rightarrow \ln d_0 \quad (41)$$

in the low-temperature limit, which is consistent with the third law of thermodynamics.

## 5.8. Transition structures

Throughout this manual we use the language of stable structures. However the code can also be applied to transition structures. A transition structure is a first-order saddle point on a transition state dividing surface that separates reactants from products. A first-order saddle point is one that has  $3N - 7$  real frequencies and one imaginary frequency, where  $N$  is the number of atoms. Sometimes, especially in the older literature, transition structures are called transition states.

Each stable structure has  $3N - 6$  real frequencies, whereas calculations for transition structures are based on the  $3N - 7$  real frequencies of each transition structure (i.e., each saddle point that is classified as a transition structure). Note that saddle points for torsional transitions (i.e., saddle points where the imaginary-frequency normal mode is the torsional coordinate) should not be called transition structures in this context; they are not used in MSTor calculations.

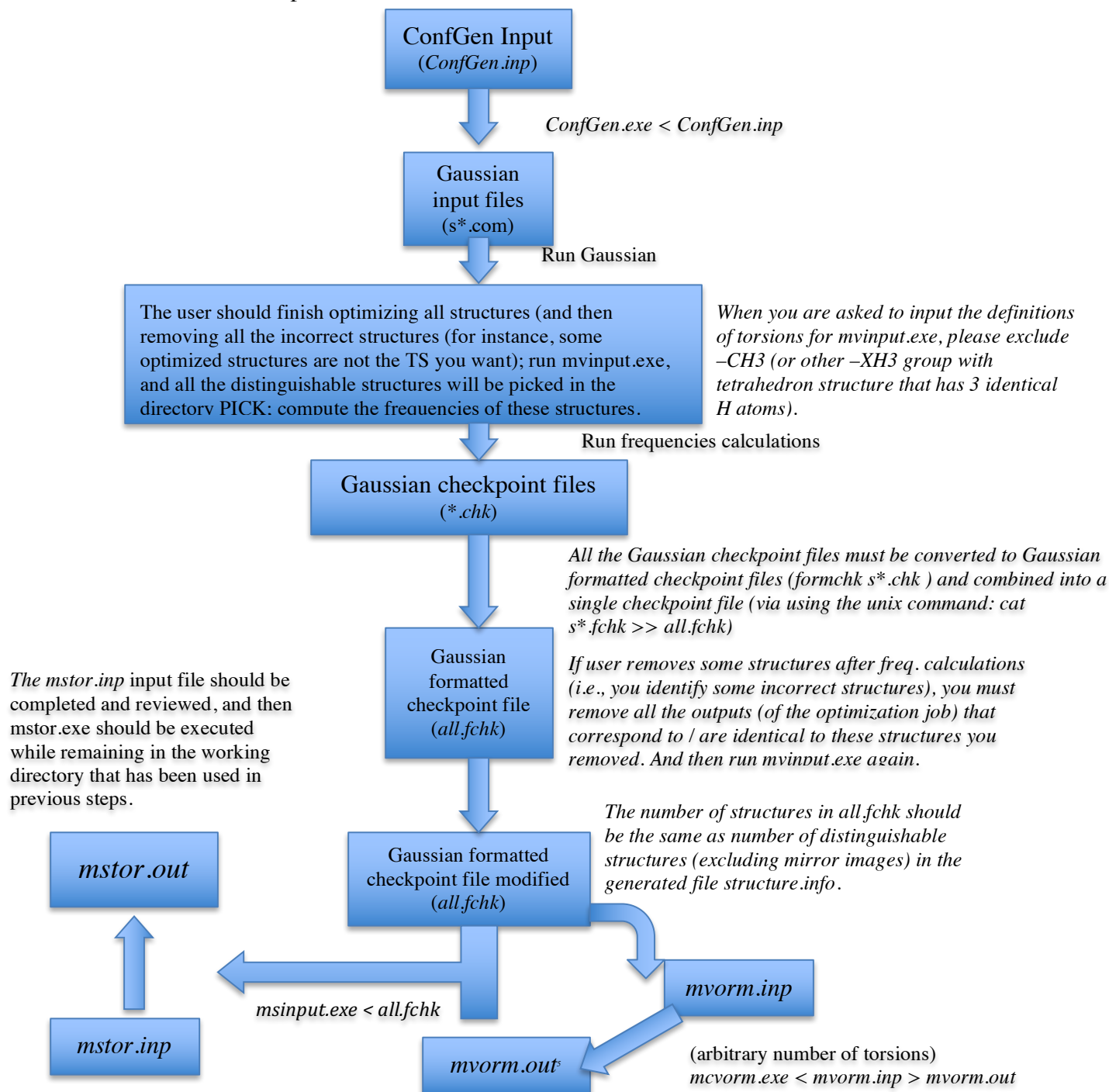
The code determines whether the system is at a stable structure or a transition-structure based on the number of imaginary frequencies.

## 5.9. Thermodynamics convention

For thermodynamic calculations based on electronic structure theory, the most convenient zero of energy is the lowest equilibrium (zero-point-exclusive) one. This choice is used throughout the manual and the code. However, almost all textbooks and compilations of thermodynamic data in the literature set the zero of energy at the zero point level. With this convention the partition function goes to the degeneracy of the ground electronic state at 0 K and enthalpy and free energy go to 0 at 0 K. For the convenience of users, we also print  $Q$ ,  $E$ ,  $H$ , and  $G$  with this thermodynamic convention. The default is to compute thermodynamic functions at 1 bar.

## 6. Guidelines to generate the *MSTor* input file with *Gaussian* output files

This section describes the steps to generate the *MSTor* input file using some utility codes in combination with the *Gaussian* software package.<sup>4</sup> The following flowchart shows one process that the user could follow:



<sup>4</sup> Frisch, M. J. *et al.* Gaussian 03/09; Gaussian, Inc.: Wallingford, CT, 2003/2009

In the first step, the user should carry out a search of all conformations that can be obtained by internal rotation of the various fragments of the initial structure. Using the Cartesian coordinates of the initial structure and the definition of each torsion angle as an input file (denoted in the flowchart as *ConfGen.inp*), the *ConfGen* utility generates the Gaussian input files of starting guesses for the different conformers and puts them in the working directory. The generated structures, obtained by all possible combinations of the values of the dihedral angles specified by the user in the input file, should then be optimized by running the *Gaussian* software package, and the output files will go in the working directory.

In the second stage, all the optimized structures should be analyzed, and the user should remove from the working directory the *Gaussian* output files for all the incorrect structures (those converged to a structure that is not what the user wanted, for instance, a torsional saddle point or a wrong transition structure). If you remove an incorrect structure, you should remove *all* the output files (from geometry optimization) that contain the identical (incorrect) structure. At this point, if all has gone well, the conformational search will have yielded all  $\tilde{J}$  structures needed for the tessellation. One of the features of the *mvinput.exe* code (which is going to be executed – see next paragraph) is that it will partially correct for deficiencies in the structure search. In particular, it will find and remove duplicate structures, and it will identify structures that should have a mirror image, and, if that mirror image is missing in the set of found structures, it will be added. Note carefully that the *mvinput.exe* code is not able to identify missing rotationally related structures that will be needed in the tessellation code so, if the system exhibits rotational symmetry, the user should be especially careful to ensure that the conformational search accounts for these needed structures.

After removing all incorrect structures from the working directory, the user should leave all remaining output files (geometry optimization files, whose file names are sXXXXXX.out, of which the Gaussian inputs are automatically generated by the conformational search utility *ConfGen.exe*) in the working directory, and next, run *mvinput.exe* by typing “*/mvinput.exe*”; this will pick out distinguishable structures (excluding mirror images or structures related by overall rotation) for you, and these output files are copied to the automatically created directory PICK. (Note that PICK is the directory containing the structures for which frequencies need to be calculated.) For example, for the case of pentane, which was discussed in detail in section 5.5, four files will be located in the PICK directory. The code will also generate an information file called *structure.info*, which includes a summary of all distinguishable structures (excluding mirror images) and all indistinguishable rotationally-related structures (excluding mirror images).

Finally, the *mvinput* utility also generates the input file, *mvorm.inp*, for the code that computes the  $M_j$  values (*mcvorm.exe* and/or *vorm.exe*). This input file will include

all  $\tilde{J}$  structures required for the tessellation. The user must always be very careful when using this file because there are some cases that deserve special attention (e.g., chiral systems). These cases are discussed in further detail in Section 7.6.

In the next step, the frequencies of the conformations that were placed in the PICK directory, (which, as stated above, include only the conformers that are distinct after fully accounting for both torsional and overall rotational symmetry and after the removal of one structure from each pair of mirror images) should be calculated and the *Gaussian* checkpoint file (including the Cartesian coordinates, the energy, and the Hessian) should be saved and subsequently formatted using the command *formchk s\*.chk*.

Once all *Gaussian* formatted checkpoint files from the frequency calculations are generated, the user must combine all of these formatted checkpoint files into a single file named all.fchk (using, for instance, the unix command: `cat s*.fchk >> all.fchk`), and this all.fchk file should be located in the same working directory as used in the second stage; this file is used by the utility called *msinput* to generate the *MSTor* input file (denoted as *mstor.inp*). The number of structures contained in all.fchk should be the same as the number of distinguishable structures (excluding mirror images) that is documented in structure.info. If, after the freq. calculation, the user removed some structures (probably because the user has identified additional incorrect structures (for instance a TS that is not what you are looking for), you need to remove *all* the output files (Gaussian geometry optimization) which contain the identical (incorrect) structures, and then run mvinput.exe again. You can check the information in structure.info to help you identify what distinguishable structures (excluding mirror images) you had in a previous run.

The automatically generated *mstor.inp* file cannot be used directly as an input file for the *MSTor* program because many parameters are not included (e.g., the definition of the non-redundant internal coordinates) and some parameters are assigned temporary values (e.g., the temperatures); also, in MS-T one should include all the torsions (including  $-\text{CH}_3$ ), but some torsions (for instance  $-\text{CH}_3$ ) are not included in mvorm.inp (i.e., not included in Voronoi tessellation because they are treated as nearly separable torsions), the user needs to add the  $M$  values of these torsions (for instance for  $-\text{CH}_3$ ,  $M=3.0$ ), and the order of the  $M$  values in mstor.inp should be consistent with the order of the definitions of the torsions in the section of the definition of the non-redundant internal coordinates in mstor.inp.

## 7. Input files

### 7.1. Input files for *mstor.exe* executable

Two input files are required for running the *mstor.exe* executable. One file contains all the information except Hessians, and the other one has the Hessians of all



structures. The former will be referred to as the main input file and the latter as the Hessian input file. Note that the name of the Hessian input file is fixed as *hess.dat*.

The main input file consists of several sections. Each section name begins with the symbol \$. At the end of each section, a keyword “END” must be given to indicate the end of this section. Each section consists of several keywords. All keywords and section names are case insensitive and any lines beginning with # symbols are comment lines.

Note that each line of the input files should be equal or less than 80 characters and characters after column 80 are ignored by the program.

### 7. 1. A. Description of sections of the main input file

<u>Section Name</u>	<u>Description</u>
<b>\$General</b>	General data to describe the system
<b>\$Intdef</b>	Defines the non-redundant internal coordinate
<b>\$Temp</b>	List of temperature
<b>\$Structure #</b>	Gives the geometry, energy, $M$ values, and weight for each structure. Here # is an integer to label the structure

### 7. 1. B. Glossary of keywords in \$GENERAL section

The section \$GENERAL is for the general information about the system, e.g., the number of atoms, the number of structures, and so on. The following keywords can be used in this section.

#### **ATM**

ATM is a keyword to use 1 atmosphere pressure for the standard state pressure. By default, if ATM is not specified, the program uses 1 bar.

#### **BAR**

BAR is a keyword to use 1 bar for the standard state pressure. This is the default, and the specification of defaults in the keyword list is optional.

#### **COUPLED**

COUPLED is a keyword to use the MS-T(C) method that is based on coupled torsional potential. This is default.

**DELTAT**

DELTAT is a keyword for specifying the temperature interval (in degrees Kelvin) of eq. (6) for calculating the heat capacity by using the four-point central finite difference formula. The default value is 1 K.

*Example:*

```
DELTAT      1
```

**ELEC**

ELEC is a keyword for specifying the degeneracy of the electronic states and their corresponding energies (in hartrees), which are used for calculating the electronic partition function. The ground electronic state has been chosen as the zero of energy. If low-lying excited states exist, they need to be included. We assume that all conformational structures have the same electronic partition function.

*Example:*

```
ELEC
  1  0.0
END
```

This example indicates that the ground electronic state is a singlet state and that this molecule has no low-lying excited states.

The example below shows a doublet ground electronic state with a low-lying doublet excited state (higher in energy than the ground state by 0.00064 hartrees).

*Example:*

```
ELEC
  2  0.0
  2  0.00064
END
```

The default, if ELEC is not specified, is “1 0.0”.

**FREQSCALE**

FREQSCALE is a keyword for specifying a frequency-scaling factor. This factor scales all the frequencies used in the calculation. The default value is 1.0. The scale factor that the user should use in thermochemistry is the ZPE scale factor, which can be found from the scale factor database (<https://comp.chem.umn.edu/freqscale/version3b2.htm>) for different electronic structure methods. When frequencies are scaled, the harmonic approximation

becomes quasiharmonic, but the results are still labeled LH because the H in LH denotes the use of harmonic formulas, not necessarily harmonic frequencies. The way that the current MSTor code does this is to read in the Hessians and scale the Hessians by the square of FREQSCALE; this is equivalent to scaling the harmonic frequencies by FREQSCALE (but this is not the way that MSTor does it).

*Example:*

```
FREQSCALE 0.970
```

## **NATOMS**

NATOMS is a keyword for specifying the number of atoms in the system. There is no default; NATOMS must be specified.

*Example:*

```
NATOMS 15
```

## **NSTR**

NSTR is a keyword for specifying the number of structures included in the calculation. This number must be equal to the number of \$Structure sections in the input file.

*Example:*

```
NSTR 15
```

There is no default; NSTR must be specified.

## **NTOR**

NTOR is a keyword for specifying the number of torsions treated with torsional corrections. Note that NTOR can be smaller than the total number of torsions in the studied molecule if one wants to treat some of them using a harmonic approximation.

*Example:*

```
NTOR 4
```

There is no default; NTOR must be specified.

## **UNCOUPLED**

UNCOUPLED is a keyword to use the MS-T(U) method that is based on an uncoupled torsional potential.

### 7. 1. C. Description of \$FRAMECHAIN section

The section \$FRAMECHAIN is an optional section that specifies how the frames in a molecule are connected to each other. The presence of this section and the (also optional) \$FRAMEDEF section that is discussed next allow KP moments of inertia to be calculated by the algorithm of Kilpatrick and Pitzer, which ensures correct results. If these sections are omitted, the det **D** is calculated from the determinant of the inverse of the  $\mathbf{G}_{\text{tor}}$  matrix, but this only yields correct results for well-chosen sets of torsional angles in the \$INTDEF section (see Section 7.1.E). Given the ease with which it is possible to select a set of torsional coordinates that does not yield a correct **D** matrix, users are strongly urged to include these optional sections. Here 1-butanol is taken as an example of the \$FRAMECHAIN input.

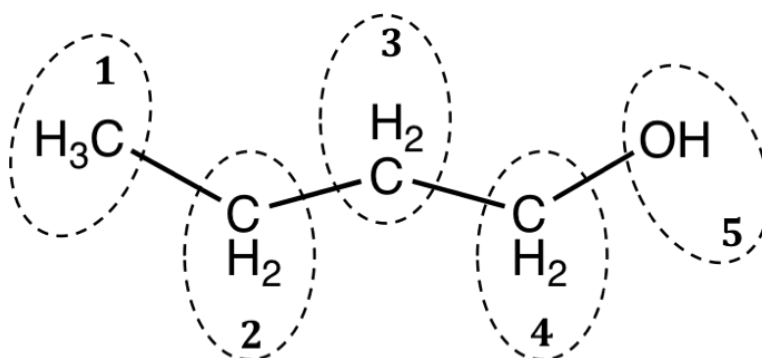


Fig 7-1. Definition of frames for 1-butanol. The group of atoms in each dashed-line circle is defined as a frame.

If frames are defined as in Fig. 7-1, we can define the \$FRAMECHAIN section as

*Example:*

```
$FRAMECHAIN
  1  2
  1  2  3
  1  2  3  4
  1  2  3  4  5
END
```

Each line in \$FRAMECHAIN begins with frame 1 and ends with the destination frame, and includes all intervening frames. One can number the frames and specify their connectivity in various ways. For example, an alternative definition of the frames for 1-butanol is given in Fig 7-2.

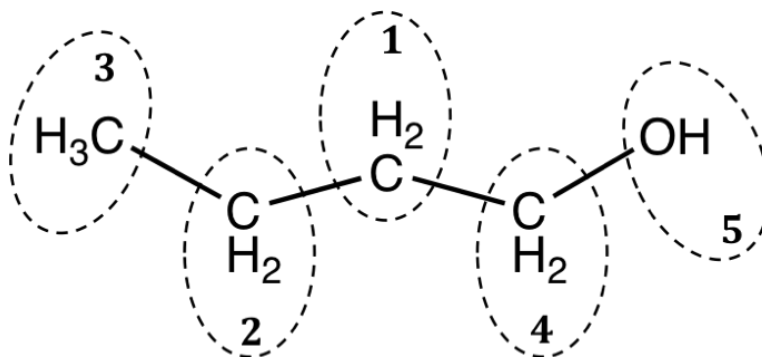


Fig. 7-2 Alternative definition of frames for 1-butanol.

If the frames are labeled as in Fig. 7-2, we can define the \$FRAMECHAIN section as

*Example:*

```
$FRAMECHAIN
  1  2
  1  2  3
  1  4
  1  4  5
END
```

### 7. 1. D. Description of \$FRAMEDEF section

The section \$FRAMEDEF is an optional section that specifies the atom numbers in each frame. When this section is used together with the also optional \$FRAMECHAIN section discussed in 7.1.C, the code will calculate the  $\det(\mathbf{D})$  using the algorithm of Kilpatrick and Pitzer, which ensures correct results. If these sections are omitted, the  $\det \mathbf{D}$  is calculated from the determinant of the inverse of the  $\mathbf{G}_{\text{tor}}$  matrix, but this only yields correct results for well-chosen sets of torsional angles in the \$INTDEF section (see Section 7.1.E) Ethanol, shown in Fig. 7-3, is taken as an example.

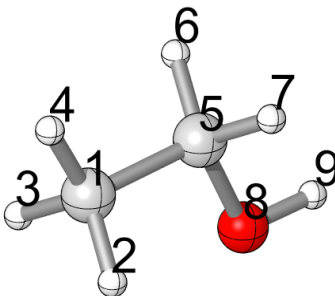


Fig. 7-3. Ethanol molecule.

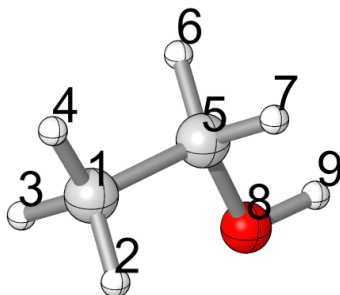
*Example:*

```
$FRAMEDEF
  1 : 1 2 3 4
  2 : 5 6 7
  3 : 8 9
END
```

In each line, the number before the colon is the frame number, and the numbers after the colon specify the atom labels of the atoms in the particular frame. Note that the colon should be separated from the numbers by spaces.

### 7. 1. E. Description of \$INTDEF section

The section \$INTDEF defines the non-redundant internal coordinates. The allowed types are stretches ( $i-j$ ), bends ( $i-j-k$ ), linear bends ( $i=j=k$ ), and dihedral angles ( $i-j-k-l$ ) where  $i, j, k$ , and  $l$  are atom labels. Each torsion should only be represented by one dihedral angle, and these dihedral angles must be at the end of the list. The following example shows how to define the non-redundant internal coordinates for the ethanol molecule shown below.



*Example:*

```
$INTDEF
  2-1 3-1 4-1 5-1 6-5 7-5 8-5 9-8
  2-1-4 2-1-3 2-1-5 3-1-4 4-1-5 1-5-6 1-5-7
  6-5-7 6-5-8 7-5-8 5-8-9
  4-1-5-8 9-8-5-1
END
```

Note that the dihedral angles for the torsions being treated in the MS-T calculation should always be put at the end of the internal coordinate list.

### 7. 1. F. Description of \$TEMP section

The section \$TEMP specifies a list of temperatures. Each line can list multiple temperatures (using integer or floating point format) and this section can have multiple lines. Note that the maximum number of temperatures allowed in this section is 100.

*Example:*

```
$TEMP
    200 298.15 300 400 600 1000
    1500 2000 2400 3000
END
```

### 7. 1. G. Description of \$STRUCTURE section

The section \$STRUCTURE specifies the information about each structure involved in the MS-T treatment. The number of \$STRUCTURE sections must be the same as that specified by the NSTR keyword in the \$GENERAL section. For each \$STRUCTURE section, a number must follow \$STRUCTURE to distinguish it from the other \$STRUCTURE sections, i.e., \$STRUCTURE 1. The following keywords can be used in these sections.

#### ENERGY

ENERGY is a keyword to specify the relative energy of the structure in kcal/mol. The global minimum is usually set to the zero of energy. Note that this energy is zero-point exclusive. The default value is zero.

*Example:*

```
ENERGY 0.077
```

#### GEOM

GEOM is a keyword to specify the Cartesian coordinates of the structure. The unit of the Cartesian coordinates is Ångstroms.

*Example:*

```
GEOM
6    -1.206324 -0.240211 -0.020995
1    -1.264318 -0.787854 -0.963519
1    -1.258873 -0.958451  0.795948
1    -2.072624  0.416915  0.056016
6     0.082734  0.557582  0.046385
1     0.126121  1.277301 -0.769964
1     0.133157  1.122438  0.982766
8     1.230304 -0.257043 -0.105397
```

```

1      1.235647 -0.918227  0.589594
END

```

Note that the first number in each line is the atomic number. For each structure, the order of the atoms should be the same.

One can specify an isotopic mass (in amu) after the coordinates for an atom. If there is nothing after the coordinates, the program uses the default mass, which is the mass of the most abundant isotope. One only needs to specify non-default isotopic masses in the first structure, and anything after the coordinates in the other structures will be ignored. Below is an example for specifying the deuterium mass in the CH<sub>3</sub>CH<sub>2</sub>OD molecule.

Note that an amu is a universal atomic mass unit equal to 1/12 the mass of a carbon-12 atom. The program uses the following conversion factor: 1 amu = 1822.888 atomic units of mass, where an atomic unit is the mass of an electron.

*Example:*

```

      GEOM
6      -1.206324 -0.240211 -0.020995
1      -1.264318 -0.787854 -0.963519
1      -1.258873 -0.958451  0.795948
1      -2.072624  0.416915  0.056016
6      0.082734  0.557582  0.046385
1      0.126121  1.277301 -0.769964
1      0.133157  1.122438  0.982766
8      1.230304 -0.257043 -0.105397
1      1.235647 -0.918227  0.589594      2.014101777
END

```

## MTOR

MTOR is a keyword to specify the  $M_{j,\tau}$  values. The order of values should be consistent with the order of the dihedral angles defined in the \$INTDEF section.

*Example:*

```

      MTOR
      3    3
END

```



If several torsional modes are strongly coupled, one could use Voronoi tessellation to determine the  $M_{j,\tau}$  values by using the utility code *vorm.exe* or *mcvorm.exe*. One runs the utility code separately and then inputs the noninteger  $M_{j,\tau}$  values here.

*Example:*

```
MTOR
  2.532  2.532    3    3
END
```

Note that there is no default value for this keyword.

## ROTSIGMA

ROTSIGMA is a keyword to specify the symmetry number of overall rotation for the structure given in the \$STRUCTURE section. The default value is 1.

*Example:*

```
ROTSIGMA 2
```

## WEIGHT

WEIGHT is a keyword to specify the number of structures that have the same energy and vibrational frequencies as that specified in GEOM. For example, if the geometry given in GEOM has a mirror image structure, one can specify a weight of 2 and only include one of the structures. The default value is 1.

*Example:*

```
WEIGHT 2
```

### 7. 1. H. Description of the *hess.dat* input file

The *hess.dat* file contains Hessians for the structures given in the \$STRUCTURE sections in the main input file. The only section name is \$HESS in *hess.dat* file. Each \$HESS should be numbered consistently with the \$STRUCTURE sections in the main input file. The Hessian should be given as in upper triangular packed form in unscaled Cartesian coordinates in hartree/bohr<sup>2</sup>. Note carefully the ordering of the matrix elements. Each line can have an arbitrary number of elements except that each line cannot exceed 80 characters.

*Example:*

```
$HESS 1
F11 F12 F22 F13 F23 F33
F14 F24 F34 F44 F15 F25
F35 F45 F55 F16 F26 F36
F46 F56 F66
END
```

## 7. 2. Input files for the *ConfGen.exe* executable

The executable *ConfGen.exe* generates a list of conformational structures by rotating a set of user-specified bonds of an input structure over a specified grid. The program writes each generated structure to a file in *Gaussian* input format.

*Example:*

```

19      2      ! number of atoms and rotating bonds
C      -10.440384 -1.363155 -2.523127 ! Cartesian coordinates
.....
# TORSION 1 DEFINITION (COMMENT LINE)
5 8      !define rotation axis as the bond between atom 5 and atom 8
7      !number of atoms in one fragment
1 2 3 14 5 6 7 ! atom numbers of this fragment
3      ! number of values that the dihedral angles take
0.0 120.0 -120.0 ! angle values by which the torsion will be rotated
#TORSION 2 DEFINITION (COMMENT LINE)
8 11     !define rotation axis as the bond between atom 8 and atom 11
10     !number of atoms in one fragment
1 2 3 14 5 6 7 8 9 10 ! atom numbers of this fragment
3      ! number of values that the dihedral angles take
0.0 120.0 -120.0 ! angle values by which the torsion will be rotated

%nproc=8      ! number of processors for Gaussian job
%mem=600mb    ! memory
# Method/Basis ! route section
0 1          ! charge and multiplicity
Extbasis      ! line at the end of the Gaussian input file for
other options (such as using external basis sets)

```

It should be noted that this utility code removes all the generated structures that show an interatomic distance smaller than a specific tolerance, currently set at 0.5 Å. (If users wish to use a different tolerance, they will need to modify the parameter "smalldist" that appears in the module "confgen\_param" listed at the top of the ConfGen.f90 file and then recompile.)

### 7.3. Input files for the *kpmoments.exe* executable

The executable *kpmoments.exe* calculates reduced moments of inertia by the method of Kilpatrick and Pitzer (KP). The input file for *kpmoments.exe* is illustrated by the following example.

*Example: (1-butanol)*

```

15 5      ! number of atoms and number of frames
0.00000000 0.00000000 0.00000000 1.00782503207 1
0.00000000 0.00000000 0.95000000 15.99491461956 1
1.32452586 0.00000000 1.40960337 12.0 2
2.11134566 1.20846054 0.96134036 12.0 3
3.52426612 1.22340239 1.50755872 12.0 4
4.31853757 2.43052798 1.05637017 12.0 5
5.32525047 2.41970290 1.46196246 1.00782503207d0 5
1.26090910 -0.01952514 2.49366282 1.00782503207d0 2
1.83970151 -0.91390364 1.10583534 1.00782503207d0 2
1.58023648 2.10625129 1.27164940 1.00782503207d0 3
2.14464785 1.23080044 -0.12912254 1.00782503207d0 3
4.03977823 0.31375819 1.20306619 1.00782503207d0 4
3.48797307 1.19751170 2.59554571 1.00782503207d0 4
3.84329672 3.35364351 1.37650469 1.00782503207d0 5
4.39964721 2.46369549 -0.02679096 1.00782503207d0 5
1 2 0 0 0 ! connectivity of frame chain
1 2 3 0 0 ! connectivity of frame chain
1 2 3 4 0 ! connectivity of frame chain
1 2 3 4 5 ! connectivity of frame chain
2 3      !start rotation axis
3 4
4 5
5 6

```

Lines 2–16 give the Cartesian coordinates (in Å) in the first three columns for each atom, the mass of each atom in the fourth column, and the label of the frame in the fifth column. Lines 17–20 give the connectivity of the frames that is explained in Sect. 7.1.C and 7.1.D. Lines 21–24 define the rotation axis for each torsion; the numbers in these lines are the numbers of the atoms, not the frame numbers.

## 7.4. Input files for the *mcvorm.exe* executable

The executable *mcvorm.exe* calculates  $M_{j,\tau}$  values defined by Voronoi tessellation of the torsional space by Monte Carlo sampling. This program can handle cases of arbitrary dimensionality.

The input file for *mcvorm.exe* is illustrated by the following example.

*Example:*

```

2      ! number of torsions (dimensions)
5      !number of structures in the input
0.005 !Monte Carlo standard error
2 1    !symmetry number for each torsion
-152.3    65.1 ! values of the dihedral angles for the first structure
159.7    178.8 ! values of the dihedral angles for the second structure
84.3    -180.0 ! values of the dihedral angles for the third structure
152.3    -65.1 ! values of the dihedral angles for the fourth structure
-159.7    -178.8 ! values of the dihedral angles for the fifth structure
```

The text after the ! symbol is an optional comment that explains the input of that line. Starting from the fifth line, each line lists the dihedral angles of the torsions for each structure.

## 7.5. Input files for the *msinput.exe* executable

The *msinput.exe* is a utility program to generate a template main input file (named *mstor.inp*) and *hess.dat* input file by extracting the needed information from *Gaussian* formatted checkpoint files. All the *Gaussian* formatted checkpoint files must be combined into a single file and this single file is the input file of the *msinput.exe* program.

Note that the generated *mstor.inp* file cannot be used directly as an input file of the *mstor.exe* program because many parameters are not assigned and some of them are assigned to some temporary numbers (e.g., the non-redundant internal coordinates are not written). The Cartesian coordinates (in Å) and relative energies (in kcal/mol with the lowest energy set to 0) in the *mstor.inp* file and the Hessians in the *hess.dat* file are taken from the checkpoint file.

If the output *mvorm.out* file from *mcvorm.exe* is given, this utility code will take the  $M_j$  values from the *mvorm.out* file (do not rename the *mvorm.out* file name). However, if the Voronoi tessellation calculation only includes a subset of the torsions, which is often the case, the user needs to input other  $M$  values for uncoupled torsions manually in the *mstor.inp* file.

This program uses the *symmetry.exe* code to determine the weight and the symmetry number of overall rotation of each structure.

## 7. 6. Running the *mvinput.exe* executable

The *mvinput.exe* is a utility program to generate a template input file, named as *mvorm.inp*, for the *vorm.exe* and/or *mcvorm.exe* programs by extracting the information from *Gaussian* optimization output files. The user will be asked to provide necessary information interactively. The number of coupled torsions included should *exclude*  $-\text{CH}_3$  groups or any other  $-\text{XY}_3$  groups, in which the X and Y are any atoms such that the group has  $\text{C}_{3v}$  symmetry. The definition of the torsions should be in the format: *a b c d*, to specify a dihedral angle between the *abc* plane and the *bcd* plane. For instance:

Please provide necessary information...

Input total number of atoms:

18

Input total number of coupled torsions:

(Excluding  $-\text{XH}_3$  (X = C, Si etc.)!!)

5

Input definitions of these torsional angles:

(Define one dihedral angle at each line)

18 17 15 16

17 15 16 5

15 16 5 13

16 5 13 14

1 5 6 9

If you have included nearly planar  $-\text{CH}_3$  groups (or, in general, nearly planar  $-\text{XY}_3$ ) in the above definition, the code will ask you to input further information. For instance, in the above definition, the first and the 4th dihedral angles are  $-\text{CH}_3$  groups. In this case you respond as follows to some additional questions:

Did you include any nearly-planar  $-\text{XH}_2$  or tetrahedron  $-\text{XH}_3$  in the above definitions?

(yes/no)

yes

Total number of  $-\text{XH}_2$  or  $-\text{XH}_3$  you included in the above definitions is?

2

And these are:

(If the j-th defined torsion is  $-\text{XH}_2$  or  $-\text{XH}_3$ , input: j)

(Define one j value at each line)

1

4

This program uses the *symmetry.exe* code to determine the symmetry point group of each structure and in certain cases, as outlined below, it adds missing mirror image

structures. If the molecule has a chiral center, no images are added. If a structure has no  $S_n$  ( $n$ -fold improper rotation axis), the values of the dihedral angles corresponding to its mirror images are included in the output file (mvorm.inp). In some unusual circumstances this can lead to two different enantiomers being present in the output file. In general, for chiral molecules, only structures corresponding to one enantiomer should be included in the tessellation calculations—even if there is sufficient energy for enantiomer interconversion—unless these enantiomers are converted via a treated torsional motion. For a molecule with a chiral atom, enantiomers cannot be interconverted via internal torsion, so only one of the enantiomers should be included in the tessellation (the one that has the same chiral-atom configuration that the user used in the initial configuration generation step). If, when asked, the user confirms the presence of a chiral atom this will automatically circumvent the addition of mirror images, which would correspond to an opposite enantiomer, by the mvinput.exe code. In some rare cases (for instance, substituted allenes), in which the molecule does not contain any chiral atom but the molecule is itself a chiral molecule, and the enantiomers can not be interconverted via internal rotations, the user should also answer “yes” to the question “Is there any chiral atom in the molecule?” when running the mvinput.exe utility, in order to prevent the code from automatically adding spurious mirror images.

The utility code does not check for missing structures that are related by an overall rotation to input structures; all such structures need to be included in the tessellation and it is the user's responsibility to make sure that they are accounted for. The user does not need to worry about this issue as long as an exhaustive conformational search is carried out.

In all cases the generated file must be checked carefully by the user before using it. For example, the symmetry numbers for all torsions in the generated file are always set to 1 and must be reset appropriately by the user.

## 7.7. Input files for the *tes.exe* executable

The *tes.exe* executable is a utility program to calculate 1-D torsional partition functions using the torsional eigenvalue summation (TES) method. The TES method is described in: B. A. Ellingson, V. A. Lynch, S. L. Mielke, and D. G. Truhlar, *Journal of Chemical Physics* **125**, 84305/1–17 (2006). Here is a brief description of the method.

A Fourier cosine series is used to represent a 1-D torsional potential

$$V = \sum_{j=0}^{j_{\max}} b_j \cos(j\phi) \quad (42)$$

where  $j_{\max}$  is the maximum order to be used,  $b_j$  is a coefficient, and  $\phi$  is torsional coordinate. The user needs to input a reduced moment of inertia and a number of potential energy values along the torsional coordinate; these values are used to fit the coefficients of the Fourier cosine series. The Hamiltonian with this fitted potential is represented in a basis of the form  $(1/\sqrt{2\pi})\exp(ik\phi)$  and diagonalized. The resulting eigenvalues are then used to calculate a partition function for a one-dimensional

separable torsion with a constant reduced moment of inertia. This algorithm is not used anywhere in a MS-T calculation; it is simply provided in case a user wishes to calculate comparison results.

The input file for the *tes.exe* is illustrated by the following example.

*Example:*

```

2
0.7456101 ! moment of inertia (in amu*Å²)
200      ! number of eigenvalues to be calculated
1.00     ! frequency scaling factor
10       ! number of terms in a Fourier cosine series to be used in potential
fitting
41       ! number of input data (angles in degree and energy in kcal/mol)
59.65955 0      ! value of the angle in degree and energy in kcal/mol
68.65955 0.053651993
77.65955 0.206199357
      :
3        ! number of temperatures
200      ! temperatures in K
800
2000

```

## 7. 8. Input files for the *vorm.exe* executable

The executable *vorm.exe* calculates  $M_j$  values using Voronoi tessellation and it can only handle 2-D and 3-D cases; for higher-dimensional systems the Monte Carlo based *mcvorm.exe* code may be used. This program calls *hull.exe* to perform the Voronoi tessellation calculations, and then reads the output of *hull.exe*.

The input file for the *vorm.exe* is illustrated by the following example.

*Example:*

```

2      ! number of torsions
5      ! number of structures
2 1    ! symmetry number for each torsion
-152.3 65.1 ! values of the dihedral angles for the first structure
159.7 178.8
84.3 -180.0
152.3 -65.1
-159.7 -178.8

```



*Explanation of the example:* The first line is the number of torsions; the second line gives the number of structures; the third line gives the symmetry number of each torsion; starting from the fourth line, each line has the dihedral angles of the torsions for each structure.

In the *mstor/utuli/vorm-test/* directory, five examples of using *vorm.exe* for calculating  $M_j$  values are provided, and their output files are located in the *mstor/utuli/vorm-test/* directory.

## 7.9 Input files for *DLMSTor.exe* executable

The generation of inputs for carrying out dual level MS-T calculations is done by the following steps.

- (1) Do a conformational search with a low-level electronic structure theory, and pick out distinguishable conformers; we name this set of conformers as set L.
- (2) Pick the first  $N$  lowest-energy conformers from set L (the value of  $N$  is based on users' choice); we name these picked conformers as set P. Re-optimize the conformers in set P with high-level electronic structure theory, and pick out distinguishable conformers; we name this set of conformers as set H.
- (3) Remove the conformers, which are in the difference set  $P \setminus H$ , from set L (here, the notation  $P \setminus H$  is the standard set theory notation, which means  $P \setminus H = \{x \in P, x \notin H\}$ ). This step removes the conformers that are distinguishable at the low level, but actually are redundant or not the correct structures at high level.
- (4) Carry out frequency calculations for conformers in set L with low-level theory (The title line of the Gaussian input must be the same as the name of the file, i.e., sXXXXXX, which is the automatically generated title for conformer XXXXXX in the conformational search step); create an empty file named LAll.fchk, and put all the sXXXXXX.fchk files of the frequency calculations into LAll.fchk.
- (5) Carry out frequency calculations for conformers in set H with high-level theory (The title line of the Gaussian input must be the same as the name of the file, i.e., sXXXXXX); create an empty file name Hpart.fchk, and put all these frequency .fchk files into Hpart.fchk.
- (6) Execute DLMSTor.exe (i.e., ./DLMSTor.exe), and the program will ask you to input the scale factor for the high-level theory and for the low-level theory, and to choose the option for reference (please input "0"); a new file named HAll.fchk is automatically generated.
- (7) In HAll.fchk, add and input the total number of coupled torsions in the first line (excluding -XH3 group (X = C, Si etc.)); and then define these torsions at each following lines. For instance, the beginning of HAll.fchk should be like (if you have 3 coupled torsions):
 

```
3
18 17 15 16
17 15 16 5
1 5 6 9
```
- (8) Proceed to compute  $M$  values, i.e., first executing mvinput.exe (./mvinput.exe), then mcvorm.exe (mcvorm.exe < mvorm.inp > mvorm.out ), and generate mstor.inp

(`msinput.exe < HAll.fchk` ), **with HAll.fchk**. After you generated `mstor.inp`, in the same directory, please execute `DLMSTor.exe` again, and this time the program will modify the `mstor.inp` file. The modified `mstor.inp` file contains “raiseL 100”, which means the frequency of the *low-level theory* conformers will be replaced by floor frequency 100  $\text{cm}^{-1}$ , if low-level computed frequency is smaller than the floor frequency. User can use other value of cutoff frequency; and we recommend the user to optimize this floor frequency especially at lower temperatures to minimize the effect of significant differences between the vibrational partition functions that are computed at high-level and low-level theory. We have proposed a formula for optimization of floor frequency in our original paper (Journal of Chemical Theory and Computation 13 (6), 2511–2522).  
 (9) Make sure that in the new `mstor.inp` file, the frequency scale factor inputted is for the *low-level theory*.  
 (10) Finally, one is ready to execute `mstor.exe`, and the results in `mstor.out` are then dual level MS-T results.

## 8. Test suites

The test suites have been designed to give some examples of input and output files for the *MSTor* program and its utility programs. The test suites include examples of radicals and transition structures as well as closed-shell molecules. The test suite is located in the `mstor/testrun/` directory, and the output files are in the `mstor/testol/` directory. There are seven directories under `mstor/testrun/` as shown below:

Directory name	For program:
ConfGen-test	<i>ConfGen.exe</i>
kpmoments-test	<i>kpmoments.exe</i>
mcvorm-test	<i>mcvorm.exe</i>
msinput-test	<i>msinput.exe</i>
mstor-test	<i>mstor.exe</i>
mvinput-test	<i>mvinput.exe</i>
tes-test	<i>tes.exe</i>
vorm-test	<i>vorm.exe</i>
DLMSTor-test	<i>DLMSTor.exe</i>

### 8.1. Test runs for *ConfGen.exe*

One input file, `2hexyl.inp`, for `ConfGen.exe` is given in the `mstor/testrun/ConfGen-test/` directory. This test run generates conformational structures for 2-hexyl radical. The initial structure is the all-trans structure. The `ConfGen.exe` program will generate 27 structures (including the initial one) by rotating three C–C bonds.

## 8.2. Test runs for *kpmoments.exe*

Two input files, *kp\_butanol.inp* and *kp\_butanol.inp*, are given in the *mstor/testrun/kpmoments-test/* directory. The two input files use the same molecule/geometry, but they label the frames differently.

## 8.3. Test runs for *mcvorm.exe*

Seven input files for the *mcvorm.exe* are given in the *mstor/testrun/mcvorm-test/* directory. They cover cases from 2D to 6D Monte Carlo sampling of the Voronoi volumes. All test runs set the target Monte Carlo standard error for *M* values to 0.005. These test runs are outlined below.

<u>Dimensions</u>	<u>Input files</u>	<u>Description</u>
2D	pentyl-2d-set1.dat	5 structures of 1-pentyl radical using a NS : SC = 2 : 2 scheme
	pentyl-2d-set2.dat	5 structures of 1-pentyl radical using NS : SC = 2 : 2.
3D	butanol.dat	29 structures of 1-butanol
	pentyl_3d.dat	15 structures of 1-pentyl radical
4D	test-4d.dat	16 evenly distributed points in 4D torsional space with each torsional coordinate going from 0 to 360 degrees.
5D	test-5d.dat	32 evenly distributed points in 5D torsional space with each torsional coordinate going from 0 to 360 degrees.
6D	test-6d.dat	64 evenly distributed points in 6D torsional space with each torsional coordinate going from 0 to 360 degrees.

## 8.4. Test runs for *msinput.exe*

One input file (*ethanol-all.fchk*) is in the *mstor/testrun/msinput-test/* directory. This input file contains two formatted checkpoint files of ethanol (two conformational structures) from *Gaussian 09* calculations. The *msinput.exe* program will generate two output files, *mstor.inp* and *hess.dat* from this run.

### 8.5. Test runs for *mvinput.exe*

One input file (*1-pentyl-all.fchk*) is in the *mstor/testrun/mvinput-test/* directory. This input file contains eight formatted checkpoint files of 1-pentyl radical (eight conformational structures) from *Gaussian 09* calculations. The *mvinput.exe* program will generate one output file, *mvorm.inp* from this run. This *mvorm.inp* file can be used as a template of the input file for *vorm.exe/mcvorm.exe* calculation.

### 8.6. Test runs for *mstor.exe*

#### 8.6.1. Ethanol

Input files for ethanol are given in the *mstor/testrun/mstor-test/ethanol/* directory.

#### 8.6.2. 1-Butanol

Input files for 1-butanol are given in the *mstor/testrun/mstor-test/butanol/* directory.

#### 8.6.3. 1-Pentyl radical

Input files for 1-pentyl are given in the *mstor/testrun/mstor-test/pentyl/* directory. Three test runs using different NS:SC schemes for  $M_{j,\tau}$  values are given in this directory. They all share the same *hess.dat* file for Hessians.

<u>Test run main input</u>	<u>Description</u>
<i>1-pentyl-intM.dat</i>	Integer $M_{j,\tau}$ values (4:0 scheme)
<i>1-pentyl-2dvor.dat</i>	$M_{j,\tau}$ values obtained by NS:SC = 2:2
<i>1-pentyl-3dvor.dat</i>	$M_{j,\tau}$ values obtained by NS:SC = 1:3.

#### 8.6.4. 1,4-Hydrogen shift saddle point of pentyl radical

Input files for 1, 4-hydrogen shift saddle point of pentyl radical are given in the *mstor/testrun/mstor-test/pentyl-ts14/* directory. There are two pairs of mirror images for the saddle point in the input file, but only the terminal methyl group torsion is treated by torsional corrections because these four structures cannot be interconverted through internal rotations.

### 8.7. Test runs for *tes.exe*

One input file, *ethanol-OH.inp*, is given in the *mstor/testrun/tes-test/* directory. This input file contains a potential of the internal rotation of the hydroxyl group in ethanol.

### 8.8. Test runs for *vorm.exe*

Five input files are given in the *mstor/testrun/vorm-test/* directory for the *vorm.exe* program. These test runs are described below.

<u>Dimensions</u>	<u>Input files</u>	<u>Description</u>
2D	pentyl-2d-set1.dat	5 structures of 1-pentyl radical using NS : SC = 2 : 2
	pentyl-2d-set2.dat	5 structures of 1-pentyl radical using NS : SC = 2 : 2
	pentyl-2d-set3.dat	5 structures of 1-pentyl radical using NS : SC = 2 : 2
3D	butanol.dat	29 structures of 1-butanol
	pentyl_3d.dat	15 structures of 1-pentyl radical using NS : SC = 1 : 3

## 9. Computers, operating systems, and compilers on which the code has been tested

### 9.1. Version 2011

Computer	OS	Compiler <sup>1</sup>
Itasca <sup>2</sup>	SuSE Linux Enterprise Server 11 SP1	ifort/11.1 and icc/11.1 gfortran/4.4 and gcc 4.4
Calhoun <sup>3</sup>	SuSE Linux Enterprise Server 10 SP3	ifort/11.1 and icc/11.1 gfortran/4.4 and gcc 4.4
Koronis <sup>4</sup>	SuSE Linux Enterprise Server 11 SP1	ifort/11.1 and icc/11.1 gfortran/4.4.3 and gcc/4.4.3
Elmo <sup>5</sup>	SuSE Linux Enterprise Server 10 SP3	ifort/11.1 and icc/11.1 gfortran/4.4.3 and gcc/4.4.3
Mac (Intel Xeon)	Mac OS X 10.5.8	gfortran/4.4 and gcc 4.4

1. On most computers two sets of compilers were tested.
2. Itasca is a Linux cluster of HP Proliant BL280c G6 Blade servers.
3. Calhoun is an SGI Altix XE 1300 cluster.
4. Koronis is a constellation of SGI systems.
5. Elmo is a Sun Fire X4600 Linux cluster.

### 9.2. Version 2011-2

Computer	OS	Compiler <sup>1</sup>
Itasca <sup>2</sup>	SuSE Linux Enterprise Server 11 SP1	ifort/11.1 and icc/11.1 gfortran/4.4 and gcc 4.4
Calhoun <sup>3</sup>	SuSE Linux Enterprise Server 10 SP3	ifort/11.1 and icc/11.1 gfortran/4.4 and gcc 4.4
Koronis <sup>4</sup>	SuSE Linux Enterprise Server 11 SP1	ifort/11.1 and icc/11.1 gfortran/4.4.3 and gcc/4.4.3
Elmo <sup>5</sup>	SuSE Linux Enterprise Server 10 SP3	ifort/11.1 and icc/11.1 gfortran/4.4.3 and gcc/4.4.3
Mac (Intel Xeon)	Mac OS X 10.5.8	gfortran/4.4 and gcc 4.4

1. On most computers two sets of compilers were tested.
2. Itasca is a Linux cluster of HP Proliant BL280c G6 Blade servers.

3. Calhoun is an SGI Altix XE 1300 cluster.
4. Koronis is a constellation of SGI systems.
5. Elmo is a Sun Fire X4600 Linux cluster.

### 9.3. Version 2011-3

Computer	OS	Compiler <sup>1</sup>
Itasca <sup>2</sup>	SuSE Linux Enterprise Server 11 SP1	ifort/11.1 and icc/11.1 gfortran/4.4 and gcc 4.4
Calhoun <sup>3</sup>	SuSE Linux Enterprise Server 10 SP3	ifort/11.1 and icc/11.1 gfortran/4.4 and gcc 4.4
Koronis <sup>4</sup>	SuSE Linux Enterprise Server 11 SP1	ifort/11.1 and icc/11.1 gfortran/4.4.3 and gcc/4.4.3
Elmo <sup>5</sup>	SuSE Linux Enterprise Server 10 SP3	ifort/11.1 and icc/11.1 gfortran/4.4.3 and gcc/4.4.3
Mac (Intel Xeon)	Mac OS X 10.5.8	gfortran/4.4 and gcc 4.4

1. On most computers two sets of compilers were tested.
2. Itasca is a Linux cluster of HP Proliant BL280c G6 Blade servers.
3. Calhoun is an SGI Altix XE 1300 cluster.
4. Koronis is a constellation of SGI systems.
5. Elmo is a Sun Fire X4600 Linux cluster.

### 9.4. Version 2013

Computer	OS	Compiler <sup>1</sup>
Itasca <sup>2</sup>	CentOS 6.2	ifort/11.1 and icc/11.1 gfortran/4.4 and gcc 4.4
Calhoun <sup>3</sup>	CentOS 6.2	ifort/11.1 and icc/11.1 gfortran/4.4 and gcc 4.4
Koronis <sup>4</sup>	SuSE Linux Enterprise Server 11 SP1	ifort/11.1 and icc/11.1 gfortran/4.4.3 and gcc/4.4.3 gfortran/4.4.3 and gcc/4.4.3
Mac (Intel Xeon)	Mac OS X 10.8.2	gfortran/4.4 and gcc 4.4

1. On most computers two sets of compilers were tested.
2. Itasca is a Linux cluster of HP Proliant BL280c G6 Blade servers.
3. Calhoun is an SGI Altix XE 1300 cluster.

4. Koronis is a constellation of SGI systems.

### **9.5. Version 2017-A and 2017-B**

Computer	OS	Compilers
Mesabi <sup>1</sup>	CentOS 6.9	ifort/13.1.3.192 and icc/13.1.3.192

---

<sup>1</sup> Mesabi is an HP Linux cluster using Intel Haswell E5-2680v3 processors.

---



## **10. Acknowledgments**

We are grateful to Serguei Patchkovskii for providing his symmetry recognition code.

This work was supported in part by the U.S. Department of Energy (DOE) Office of Science, Office of Basic Energy Sciences, as part of the Combustion Energy Frontier Research Center under Award Number DE-SC0001198. This work was also supported by the DOE through grant no. DE-FG02-86ER13579.

## 11. Revision history

### **Version 2011 (October 6, 2011)**

Authors: J. Zheng, S. L. Mielke, K. L. Clarkson, and D. G. Truhlar

This is the first distributed version.

### **Version 2011-2 (November 2, 2011)**

Authors: J. Zheng, S. L. Mielke, K. L. Clarkson, and D. G. Truhlar

This revision adds the capability to calculate the heat capacity at constant pressure by using a four-point central finite difference scheme.

In addition, more comments were added to the source code.

### **Version 2011-3 (February 15, 2011)**

Authors: J. Zheng, S. L. Mielke, K. L. Clarkson, and D. G. Truhlar

This is the first version distributed by *CPC*.

This revision adds one more utility programs, in particular a program to calculate 1-D torsional partition functions using the torsional eigenvalue summation (TES) method.

### **Version 2011-3-A (December 19, 2012)**

Authors: J. Zheng, S. L. Mielke, K. L. Clarkson, and D. G. Truhlar

Contributors to this revision: J. Zheng and D. G. Truhlar

This revision has a bug fixed in the vib.f90 file. Line 33 is uncommented to initialize the array of center-of-mass coordinates as zero.

### Version 2013 (January 31, 2013)

Authors: J. Zheng, S. L. Mielke, K. L. Clarkson, R. Meana-Pañeda, and D. G. Truhlar

Contributors to new revision: J. Zheng, R. Meana-Pañeda, and D. G. Truhlar

This revision is submitted to *Computer Physics Communication* in February 2013 by means of a new version announcement.

New features and bug fixes:

- The MS-T method based on a coupled torsional potential is added, which is called MS-T(C), where C denotes coupled. The original method that is based on an uncoupled potential is called MS-T(U).
- The capability of treating linear bending motions is added.
- The sections \$framechain and \$framedef are no longer needed in the input file, which simplifies the input file. (Note that these sections were reintroduced into the code as optional, but strongly recommended, sections in version 2017.)
- The code for evaluating of the point group symmetry of a structure from Serguei Patchkovskii is included.
- The *mvinput.exe* utility program is added to generate the input file for the *vorm.exe* and/or *mcvorm.exe* codes. The *msinput.exe* utility code has been modified to write the *M* values taken from the *mvorm.out* file. Both *mvinput.exe* and *msinput.exe* use the *symmetry.exe* program to determine the point group of each structure.
- The *ConfGen.exe* has been completely written in Fortran 90 and reads news fields corresponding to the Gaussian input keywords.
- A bug in the generation of different structures that affects *ConfGen.exe*, *vorm.exe* and *mcvorm.exe* utility codes has been fixed

### Version 2017 (June 23, 2017)

Authors: J. Zheng, S. L. Mielke, J. L. Bao, R. Meana-Pañeda, K. L. Clarkson, and D. G. Truhlar

Contributors to new revision: J. L. Bao, S. L. Mielke, and D. G. Truhlar

New features and bug fixes:

- A new utility code, DLMSTor.exe, was provided to allow dual level calculations.
- The \$framechain and \$framedef sections were reintroduced to the code as optional sections. If these sections are used, the **D** matrix is calculated using the scheme of Kilpatrick and Pitzer and one should always get correct values for the KP moments of inertia. If the user omits these sections, the **D** matrix is calculated from the inverse of the  $G_{\text{tor}}$  matrix, but one must choose the torsional coordinates carefully to ensure that this produces correct results. Use of the \$framechain and \$framedef sections is recommended. This option is explained in sections 7.1.C and 7.1.D of the manual.
- Various utility codes were modified to help ensure that structures that are equivalent under rotation to other structures (and thus indistinguishable when considering overall rotation and torsion) are still properly included in the Voronoi tessellation step (which does not fully exploit overall rotational symmetry).
- The mcvorm.exe utility code was modified to correctly calculate uncertainties and to take advantage of rotational symmetries (at least for linear chain molecules) and mirror-image symmetries, if present, in the calculation of the uncertainty estimates.
- The following sections of the manual are greatly improved:
  - 5.5. Treatment of rotational symmetry and mirror images
  - 5.6. Voronoi tessellation scheme for  $M$  values

**Version 2017-A (Aug 02, 2017)**

Fixing minor bugs that were in the input.f90 file that affected the counting of the number of degree of freedoms for linear bending modes.

**Version 2017-B (Nov. 20, 2017)**

Minor improvement for mvinput.f90 is updated.