TUMME 3.0

MANUAL

Rui Ming Zhang

Center for Combustion Energy, Department of Energy and Power Engineering, and Key Laboratory for Thermal Science and Power Engineering of Ministry of Education, Tsinghua University, Beijing 100084, China

and Department of Chemistry, Chemical Theory Center, and Minnesota Supercomputing Institute, University of Minnesota, Minneapolis, Minnesota 55455-0431, USA

Xuefei Xu

Center for Combustion Energy, Department of Energy and Power Engineering, and Key Laboratory for Thermal Science and Power Engineering of Ministry of Education, Tsinghua University, Beijing 100084, China

and

Donald G. Truhlar

Department of Chemistry, Chemical Theory Center, and Minnesota Supercomputing Institute, University of Minnesota, Minneapolis, Minnesota 55455-0431, USA

Date of final update to code: July 6, 2022
Date of most recent revision to this manual: July 6, 2022

ABSTRACT. TUMME (Tsinghua University Minnesota Master Equation solver) is a computer program for setting up and solving one-dimensional energy-dependent master equations for chemical kinetics of unimolecular and bimolecular reactions. For mechanisms involving only unimolecular isomerization (no bimolecular pairs), TUMME solves a conservative master equation for both rate constants and time-dependent energy-bin populations. For mechanisms involving bimolecular pairs, TUMME solves two kinds of master equation: (i) a nonconservative master equation for calculating rate constants of bimolecular reactions and (ii) a conservative master equation that includes bimolecular association in the transition matrix for calculating the time-evolution of the concentration of a pseudo-first-order bimolecular reactant. TUMME has interfaces to *Gaussian*, *Polyrate*, and/or *MSTor* output files that allow the master equation code to provide the microcanonical flux coefficients needed for the kernel of the ME as calculated by multi-structural variational transition state theory with small-curvature tunneling (MS-VTST/SCT). TUMME is written in double precision with Python 3; quadruple and octuple precision is also available for some subtasks in C++. The Python code can run in serial or parallel (MP or MPI), and C++ code can run on a single processor or on multiple processors with OpenMP.

Reference to the program: R. M. Zhang, X. Xu, and D. G. Truhlar, TUMME *3.0*, https://comp.chem.umn.edu/tumme

Reference to the article describing the program: R. M. Zhang, X. Xu, and D. G. Truhlar, "TUMME: *Tsinghua University Minnesota Master Equation* program," Computer Physics Communications, **2022,** *270*, 108140. https://doi.org/10.1016/j.cpc.2021.108140

Table of Contents

1. Introduction

The time development of a chemical reaction mechanism involving several species and/or several states can be approximately described as a stochastic process and in particular as a Markov chain. It has long been recognized that the time evolution of species concentration according to a give mechanism can be simulated by a multi-state master equation, and that chemical reaction rate constants can be extracted by eigenanalysis of such a master equation.[1] A master equation is a set of coupled ordinary differential equations with time as the independent variable. If all chemical species are at internal equilibrium, the dependent variables are the concentrations of the chemical species. Of more interest here is the case in which internal-state nonequilibrium is allowed, then the independent variables are the concentrations of chemical species in individual internal states or, more commonly, the concentrations of chemical species summed over bins of internal states. An especially powerful method, applicable to all unimolecular processes and under many conditions extendable to second-order reactions, is to linearize the master equation in the various concentrations so the eigenvalues have units (s^{-1}) of unimolecular rate constants or pseudo first-order rate constants, and the rate constants can be related to the slowest eigenvalues.[2]

Because the master equation is especially powerful for modeling nonequilibrium effects, and because nonequilibrium effects are more important in unimolecular reactions than most bimolecular reactions (with bimolecular association reactions being the exception because they are the reverse of unimolecular reactions), master equations have found their main use in unimolecular reactions. Most modern theoretical treatments of unimolecular reactions make the Rice-Ramsperger-Kassel assumption[3,4] that the rate of a unimolecular reaction depends only on the total energy content (E) of the molecule. In this case the internal-state bins of each species are specified by a single variable E, and the treatment is sometimes called a 1-D, energy-dependent master equation. TUMME is a computer program for setting up and solving 1-D, energy-dependent master equations. In particular, the master equation is discretized in terms of chemical species in finite-width internal energy bins. In the present context, internal energy is the total vibrational–rotational energy. We note that that some workers label the bins as grains or intervals.

The extraction of chemical kinetics rate constants by eigenanalysis of 1-D, energy-dependent master equations has come to be known as the method of chemically significant eigenmodes (CSE theory), and its use for the treatment of complex mechanisms has been greatly developed and clarified in recent years; we will build on this work, in particular using the method of Georgievskii et al.[5] This method is applicable to unimolecular isomerization and dissociation reactions proceeding from one or multiple isomers. (In this manual, well and isomer are synonyms.) For the case where the concentrations of the bimolecular reactants do not change significantly on the relaxation time scale, association reactions from bimolecular pairs can also be considered. The resulting rate constants depend on temperature T and pressure p.

This program is written in Python 3 utilizing C++ high-precision dynamic libraries. For running multiple combinations of T and p, the execution can be parallelized on a single node using MP or across multiple nodes using MPI. The program can be run in double precision, quadruple, or octuple precision.

A key feature of the TUMME program is that it is interfaced to the *Polyrate* program for calculating rate constants and to the *MSTor* program for the treatment of torsional anharmonicity. By reading the minimum energy path and the transmission possibilities from *Polyrate* and reading the multi-structural torsional density of states from *MSTor*, this program can use pressure-independent MS-VTST/SCT rate constants for elementary reactions as input data for the pressure-dependent master equation solver for complex reaction systems.

2. Theory

TUMME can treat both unimolecular reactions and bimolecular reactions, but bimolecular reactions are treated only in the limit of pseudo-first-order kinetics. TUMME can treat two kinds of master equation, with the difference being the way that one treats the bimolecular pairs. One kind of master equation is conservative (also called reversible or homogeneous); this kind of master equation leads to equilibrium. The other kind of master equation is nonconservative (also called irreversible or inhomogeneous); for mechanisms involving unimolecular dissociation, this kind of master equation leads to all isomers vanishing because one includes the dissociation reactions but not the reverse association reactions. If there are no bimolecular species in the mechanism, the master equation is always conservative. For mechanisms involving bimolecular pairs, TUMME *uses the nonconservative master equation to extract phenomenological rate constants and uses the conservative master equation to calculate the time evolution of concentrations and energy-bin populations.*

Here we only give a brief review of the theory. For more details, the user should refer to ref. 25 for the case of only unimolecular reactions and to ref. 27 for the case where bimolecular reactions are included.

2.1 Master equation

If we collect microscopic concentration of energy bins of isomers into one vector **y** (with a dimension of N_y) and treat bimolecular pairs as irreversible sinks, we can write the discrete master equation as an inhomogeneous equation in the following way

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = -\mathbf{W}\mathbf{y} + \mathbf{B}\mathbf{s} \tag{1}$$

where t is time, **s** is the macroscopic concentration vector of bimolecular pairs, **B** is the association reaction flux coefficient matrix, and **W** is the transition matrix given by

$$\mathbf{W} = \widehat{\mathbf{K}} + \mathbf{P} \tag{2}$$

where $\widehat{\mathbf{K}}$ is the unimolecular chemical reaction flux coefficient matrix including isomerization and dissociation (the caret is used to distinguish reactive flux coefficients, which will have carets, from rate constants, which will not), and **P** is the collisional energy-relaxation matrix. We transform **W** to a symmetric matrix **G** with positive eigenvalues according to

$$\mathbf{G} = \mathbf{F}^{-1}\mathbf{W}\mathbf{F} \tag{3}$$

where **F** is a diagonal matrix with diagonal elements

$$F_i = \sqrt{\rho_\gamma(E_\eta)\exp(-\beta E_\eta)} \tag{4}$$

where i is the state index; γ is the index of isomers; η is the index of each energy bin; ρ is the electronic-rotational-vibrational density of state; $\beta$is $1/\mathrm{k_B}T$. This transformation leads to the following symmetrized master equation

$$\frac{\mathrm{d}\tilde{\mathbf{y}}}{\mathrm{d}t} = -\mathbf{G}\tilde{\mathbf{y}} + \widetilde{\mathbf{B}}\mathbf{s} \tag{5}$$

with $\tilde{\mathbf{y}} = \mathbf{F}^{-1}\mathbf{y}$ and $\widetilde{\mathbf{B}} = \mathbf{F}^{-1}\mathbf{B}$.

2.1.1 Collisional relaxation energy transfer possibility

In TUMME, we use the "exponential-down" model to describe the relaxation of isomers by collisions with a bath gas; this model assumes that the probability of a collision changing the energy of isomer γ from initial energy $E_{\eta'}$ to final energy E_η is

$$P_\gamma(\eta|\eta') = \begin{cases} A(E_{\eta'})e^{-\frac{E_{\eta'}-E_\eta}{\langle\Delta E_d\rangle(E_{\eta'})}} & \text{for } E_{\eta'} \geq E_\eta \\ \left[A(E_\eta)\, e^{-\frac{E_\eta-E_{\eta'}}{\langle\Delta E_d\rangle(E_\eta)}}\right] \frac{\rho_\gamma(E_\eta)e^{-\beta E_\eta}}{\rho_\gamma(E_{\eta'})e^{-\beta\eta'}} & \text{for } E_{\eta'} < E_\eta \end{cases} \quad (6)$$

2.1.2 Microcanonical flux coefficients

In the following part, we use ρ to denote the electronic-rotational-vibrational density of states, of which the Laplace transform is the electronic-rotational-vibrational partition function (Q_ϕ), and we use ψ to denote the electronic-rotational-vibrational density of states including relative translation, of which the Laplace transform is the electronic-rotational-vibrational partition function (Q_ϕ) times the relative translational partition function per unit volume (Φ_{rel}).

(1) Based on the framework of RRKM theory

Unimolecular reaction. For a unimolecular reaction, the microcanonical flux coefficient at energy E_η from configuration γ to configuration ϕ (an isomer or a bimolecular pair) is:

$$\hat{k}^{\text{MS-VTST/SCT}}\left(\gamma \to \phi \mid E_\eta\right) = \kappa^{\text{SCT}}(E_\eta)\Gamma^{\text{VTST}}(E_\eta)F^{\text{MS}}(E_\eta)\frac{N^{\ddagger,\text{SSHO}}_{\gamma\to\phi}(E_\eta)}{h\rho^{\text{SSHO}}_\gamma(E_\eta)} \quad (7)$$

where MS denotes that multi-structural effects and/or torsional anharmonicity are included; VTST denotes variational transition state theory, which may be canonical variational theory (CVT) or microcanonical variational theory (μVT); SCT denotes small-curvature tunneling; $\kappa^{\text{SCT}}, \Gamma^{\text{VTST}},$ and F^{MS} are transmission coefficients; SSHO denotes single-structural harmonic oscillator or single-structural quasiharmonic oscillator; and a diesis (‡) denotes a conventional transition state, i.e., that the transition state properties are evaluated at a dividing surface passing through a saddle point on the potential energy surface. N denotes sum of states or cumulative reaction possibility. ρ denotes the electronic-rotational-vibrational density of state.

The transmission coefficients are approximated as follows. The microcanonical MS anharmonicity coefficient is evaluated by

$$F^{\text{MS}}(E_\eta) = \frac{N^{\ddagger,\text{MS}}_{\gamma\to\phi}(E_\eta)}{N^{\ddagger,\text{SSHO}}_{\gamma\to\phi}(E_\eta)}\frac{\rho^{\text{SSHO}}_\gamma(E_\eta)}{\rho^{\text{MS}}_\gamma(E_\eta)} \quad (8)$$

the microcanonical tunneling transmission coefficient is evaluated by

$$\kappa^{\text{SCT}}(E_\eta) = \frac{N^{\ddagger,\text{MS/SCT}}_{\gamma\to\phi}(E_\eta)}{N^{\ddagger,\text{MS}}_{\gamma\to\phi}(E_\eta)} \quad (9)$$

and the microcanonical recrossing transmission coefficient is approximated evaluated by

$$\Gamma^{\text{VTST}}(E_\eta) \approx \frac{N^{\text{VTS,SSHO/SCT}}_{\gamma\to\phi}(E_\eta)}{N^{\ddagger,\text{SSHO/SCT}}_{\gamma\to\phi}(E_\eta)} \quad (10)$$

Note that the numerator of Eq. (10) is evaluated at a variational transition state, and the denominator is evaluated at a conventional transition state. When $N_{\gamma\to\phi}(E_\eta)$ is marked with

tunneling superscript in Eqs. (9) and (10), it is the cumulative reaction probability (CRP); otherwise, it is the sum of states (SoS). When CVT specified, for each temperature, TUMME will choose a variational transition state along the reaction coordinate according to the maximum Gibbs free energy barrier. When μVT specified, for each energy bin, the program will choose a variational transition state along the reaction coordinate according to the minimum SoS.

The cumulative reaction probability is evaluated as the convolution of the transmission probability with the density of states of the transition state:

$$N_{\gamma\to\phi}^{\mathrm{VTS,MS/SCT}}(E_\eta) = \int_{\max\{E_{0,\gamma},E_{0,\phi}\}-V_a^{G^*}}^{E_\eta-V_a^{G^*}} \mathrm{d}E P_{\gamma\to\phi}^{\mathrm{SCT}}(E)\rho_\gamma^{\mathrm{VTS,MS}}(E_\eta - E) \tag{11}$$

where E is the energy (which can be negative) in the reaction coordinate at the VTS, $P_{\gamma\to\phi}^{\mathrm{SCT}}$ is the small-curvature tunneling (SCT) approximation to the transmission probability of the reaction between γ and ϕ, $\rho_{\gamma\to\phi}^{\mathrm{VTS,MS}}$ is the multi-structural torsional electronical-vibrational-rotational density of states of the variational transition state connecting γ and ϕ, and E_0^{VTS}, $E_{0,\gamma}$ and $E_{0,\phi}$ are the enthalpy respectively of the variational transition state, of isomer γ, and of isomer or bimolecular pair ϕ at 0 K. The transmission probability $P_{\gamma\to\phi}^{\mathrm{SCT}}$ is evaluated by the SCT method implemented in *Polyrate*. The multi-structural density of states $\rho_\gamma^{\mathrm{VTS,MS}}$ is evaluated as the inverse Laplace transform of the multi-structural torsional partition function with the first-order steepest descent method implemented in *MSTor*[6 7 8].

Bimolecular reaction. For a bimolecular reaction, we use the general microcanonical flux coefficients $\Delta\hat{k}(E)$ instead of $\hat{k}(E)$; $\Delta\hat{k}(E)$ is the following product of $\hat{k}(E)$ and the Boltzmann distribution:

$$\Delta\hat{k}^{\mathrm{MS-VTST/SCT}}\left(\nu\to\gamma|\,E_\eta\right) = \Gamma^{\mathrm{VTST}}(E_\eta)\kappa^{\mathrm{SCT}}(E_\eta)F^{\mathrm{MS}}(E_\eta)\frac{N_{\nu,\gamma}^{\ddagger,\mathrm{SSHO}}(E_\eta)e^{-\beta E_\eta}\Delta E}{h\Phi_{\mathrm{rel}}Q_\nu^{\mathrm{SSHO}}} \tag{12}$$

where ΔE is the width of an energy bin, Φ_{rel} denotes relative translational partition function per unit volume, $\Gamma^{\mathrm{VTST}}(E_\eta)$ and $\kappa^{\mathrm{SCT}}(E_\eta)$ are the same as in Eqs (9) and (10), and $F^{\mathrm{MS}}(E_\eta)$ is defined as

$$F^{\mathrm{MS}}(E_\eta) = \frac{N_{\nu,\gamma}^{\ddagger,\mathrm{MS}}(E_\eta)}{N_{\nu,\gamma}^{\ddagger,\mathrm{SSHO}}(E_\eta)}\frac{Q_\nu^{\mathrm{SSHO}}(E_\eta)}{Q_\nu^{\mathrm{MS}}(E_\eta)} \tag{13}$$

(2) Based on the inverse-Laplace transform

TUMME also provides an option for user to calculate the microcanonical flux coefficients according to the inverse-Laplace transform to the canonical flux coefficients (high-pressure-limit rate constant).

Unimolecular reaction. For a unimolecular reaction $\gamma\to\phi$, the canonical flux coefficients are

$$\hat{k}(\gamma\to\phi|T) = \frac{1}{Q_\gamma}\int_0^\infty \hat{k}(\gamma\to\phi|E)\rho_\gamma(E)e^{-\beta E}dE = \frac{1}{Q_\gamma}\mathcal{L}\left[\hat{k}(\gamma\to\phi|E)\rho_\gamma(E)\right] \tag{14}$$

Thus,

$$\hat{k}(\gamma\to\phi|E) = \frac{1}{\rho_\gamma(E)}\mathcal{L}^{-1}\left[\hat{k}(\gamma\to\phi|T)Q_\gamma\right](E) \tag{15}$$

We fit the canonical flux coefficients by a biexponential form as

$$\hat{k}(\gamma \to \phi|T) = A_1\left(\frac{\beta_1}{\beta}\right)^{n_1} e^{-\beta E_{a_1}} + A_2\left(\frac{\beta_2}{\beta}\right)^{n_2} e^{-\beta E_{a_2}} \tag{16}$$

One can get the microcanonical flux coefficients for $\gamma \to \phi$ according to Eqs 错误!未找到引用源。(15) and (16) as

$$\begin{aligned}\hat{k}(\gamma \to \phi|E_\eta) = &\frac{A_1\beta_1^{n_1}}{\rho_\gamma(E_\eta)\Gamma(n_1)}\int_0^{E_\eta}\rho_\gamma(E_\eta-\varepsilon)(\varepsilon-E_{a_1})^{n_1-1}\theta(\varepsilon-E_{a_1})\,\mathrm{d}\varepsilon \\ &+\frac{A_1\beta_1^{n_2}}{\rho_\gamma(E_\eta)\Gamma(n_2)}\int_0^{E_\eta}\rho_\gamma(E_\eta-\varepsilon)(\varepsilon-E_{a_2})^{n_2-1}\theta(\varepsilon-E_{a_2})\,\mathrm{d}\varepsilon\end{aligned} \tag{17}$$

Bimolecular reaction. For the bimolecular reaction $\nu \to \gamma$, the canonical flux coefficients are

$$\hat{k}(\nu \to \gamma|T) = \frac{1}{\Phi_{\mathrm{rel}}Q_\nu}\int_0^\infty \hat{k}(\nu \to \gamma|E)\psi_\nu(E)e^{-\beta E}\mathrm{d}E = \frac{1}{\Phi_{\mathrm{rel}}Q_\nu}\mathcal{L}\left[\hat{k}(\nu \to \gamma|E)\psi_\nu(E)\right] \tag{18}$$

The relative translational partition function is

$$\Phi_{\mathrm{rel}} = \left(\frac{2\pi m_\nu}{h^2\beta}\right)^{3/2} \tag{19}$$

where m_ν is the reduced mass. Thus,

$$\hat{k}(\nu \to \gamma|E) = \frac{1}{\psi_\nu(E)}\mathcal{L}^{-1}[\hat{k}(\nu \to \gamma|T)\Phi_{\mathrm{rel}}Q_\nu](E) \tag{20}$$

We fit the canonical flux coefficients by a biexponential form as

$$\hat{k}(\nu \to \gamma|T) = A_1\left(\frac{\beta_1}{\beta}\right)^{n_1} e^{-\beta E_{a_1}} + A_2\left(\frac{\beta_2}{\beta}\right)^{n_2} e^{-\beta E_{a_2}} \tag{21}$$

One can get the microcanonical flux coefficients for $\nu \to \gamma$ according to Eqs 错误!未找到引用源。(19), (20) and (21) as

$$\begin{aligned}\hat{k}(\nu \to \gamma|E_\eta) = &\frac{\frac{1}{h^3}(2\pi m_\nu)^{\frac{3}{2}}A_1\beta_1^{n_1}}{\psi_\nu(E_\eta)\Gamma\left(n_1+\frac{3}{2}\right)}\int_0^{E_\eta}\rho_\nu(E_\eta-\varepsilon)(\varepsilon-E_{a_1})^{n_1+\frac{1}{2}}\theta(\varepsilon-E_{a_1})\,\mathrm{d}\varepsilon \\ &+\frac{\frac{1}{h^3}(2\pi m_\nu)^{3/2}A_2\beta_2^{n_2}}{\psi_\nu(E_\eta)\Gamma\left(n_2+\frac{3}{2}\right)}\int_0^{E_\eta}\rho_\nu(E_\eta-\varepsilon)(\varepsilon-E_{a_2})^{n_2+\frac{1}{2}}\theta(\varepsilon-E_{a_2})\,\mathrm{d}\varepsilon\end{aligned} \tag{22}$$

and the general microcanonical flux coefficients

$$\begin{aligned}\Delta\hat{k}(\nu \to \gamma|E_\eta) = &\frac{e^{-\beta E_\eta}\Delta E}{\Phi_{\mathrm{rel}}Q_\nu^{\mathrm{SSHO}}}\left[\frac{\frac{1}{h^3}(2\pi m_\nu)^{\frac{3}{2}}A_1\beta_1^{n_1}}{\Gamma\left(n_1+\frac{3}{2}\right)}\int_0^{E_\eta}\rho_\nu(E_\eta-\varepsilon)(\varepsilon-E_{a_1})^{n_1+\frac{1}{2}}\theta(\varepsilon-E_{a_1})\,\mathrm{d}\varepsilon\right. \\ &\left.+\frac{\frac{1}{h^3}(2\pi m_\nu)^{3/2}A_2\beta_2^{n_2}}{\Gamma\left(n_2+\frac{3}{2}\right)}\int_0^{E_\eta}\rho_\nu(E_\eta-\varepsilon)(\varepsilon-E_{a_2})^{n_2+\frac{1}{2}}\theta(\varepsilon-E_{a_2})\,\mathrm{d}\varepsilon\right]\end{aligned} \tag{23}$$

2.2 Phenomenological Rate Constant

The phenomenological rate constants are extracted from the inhomogeneous (irreversible) master equation (5) based on CSE theory. The derivation is complicated and lengthy, so we do not present here; users could refer to ref. 25 for the formula and derivation.

2.3 Time evolution

The irreversible master equation (5) cannot be solved directly since the bimolecular concentrations are unknown. Here in order to introduce more equations to describe the source term, we partition bimolecular pairs into two kinds, the reversible reactants and the irreversible products. For the bimolecular reactant, we introduce the pseudo-first order assumption; for the bimolecular product, we assume the bimolecular concentration constantly as zero. Supposing there are *r* reactant bimolecular pairs, thus the Eq (1) becomes

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = -\mathbf{W}\mathbf{y} + \mathbf{B}^*\mathbf{s}^* \tag{24}$$

where $\mathbf{B}^*$ and $\mathbf{s}^*$only contain *r* bimolecular pairs and each one bimolecular pair (A + B) has one control equation as

$$\frac{\mathrm{d}n_{\mathrm{A}}^{(\nu)}}{\mathrm{d}t} = -\left(n_{\mathrm{B}}^{(\nu)}\Sigma_i B_{i\nu}^*\right)n_{\mathrm{A}}^{(\nu)} + \Sigma_i \hat{k}(\gamma \to \nu \mid E_\eta)y_i \tag{25}$$

where the index ν ranges over 1 to *r*, and the macroscopic concentration $n_{\mathrm{B}}^{(\nu)}$ is assumed to be in excess as compared to the macroscopic concentration $n_{\mathrm{A}}^{(\nu)}$ and taken as a constant. Combined with Eq (25), the inhomogeneous master equation (24) can be transformed into the following homogeneous master equation:

$$\frac{\mathrm{d}\mathbf{y}^*}{\mathrm{d}t} = -\mathbf{W}^*\mathbf{y}^* \tag{26}$$

where

$$y_i^* = \begin{cases} y_i & 0 < i \le N_{\mathrm{y}} \\ n_{\mathrm{A}}^{(\nu)} & N_{\mathrm{y}} < i \le N_{\mathrm{y}} + r \end{cases} \tag{27}$$

Analogy to Eq (5), Eq (26) can be symmetrized as

$$\frac{\mathrm{d}\tilde{\mathbf{y}}^*}{\mathrm{d}t} = -\mathbf{G}^*\tilde{\mathbf{y}}^* \tag{28}$$

by a diagonal matrix $\mathbf{F}^*$ with elements as

$$F_{ii}^* = \begin{cases} \sqrt{\rho_\gamma(E_\eta)\exp(-\beta E_\eta)}; & 0 < i \le N_{\mathrm{y}} \\ \sqrt{\dfrac{\hat{k}(\gamma' \to \nu \mid E_{\eta'})}{B_{i'\nu}n_{\mathrm{B}}^{(\nu)}}\rho_{\gamma'}(E_{\eta'})e^{-\beta E_{\eta'}}}\ ; & N_{\mathrm{y}} < i \le N_{\mathrm{y}} + r \end{cases} \tag{29}$$

where the index i' is an arbitrary value among 1 and N_{y}. The result can be solved as

$$\mathbf{y}^*(t) = \mathbf{F}^*\mathbf{U}^*\mathbf{E}^*(\mathbf{U}^*)^{\mathrm{T}}(\mathbf{F}^*)^{-1}\mathbf{y}_0^* \tag{30}$$

where $\mathbf{U}^*$ is the eigenvector matrix of symmetric transition matrix $\mathbf{G}^*$, $\mathbf{E}^*$ is a diagonal matrix with elements as $e^{-L_\lambda^* t}$ where L_λ^* is the eigenvalue of matrix $\mathbf{G}^*$.

3. Program Description

3.1 General workflow

TUMME is written in Python 3. It can run in three different modes: serial, multi-process (MP) and message passing interface (MPI). Serial mode uses a single processor, whereas the latter two modes involve parallel execution on multiple processors by running various temperature–pressure pairs (*T, p*) simultaneously. The workflow is depicted in Fig. 1. The input file is described in Section 3. The general workflow can be summarized as follows.

1) First, the program reads a standard input file containing information about the species and the reactions in the mechanism and a variable determining the mode in which to execute the program. If assigned in the standard input file, some output files of *Polyrate* will be read. This part is done in *tumme_readin.py.*
2) After reading all the input information, the program will enter the initialization module to initialize (pre-set) some variables and to calculate the density of states. Alternatively, depending on settings in the input, the density of states of some species may be read from the *MSTor* output file. This part is done in *tumme_pre.py.*
3) Then the program will jump into the solver module shown as the standard process in Fig. 1. From this point on, the precision (double, quadruple or octuple) and mode (serial, MP, or MPI) may vary according to users' choice. This part is done in *tumme_solver.py*.

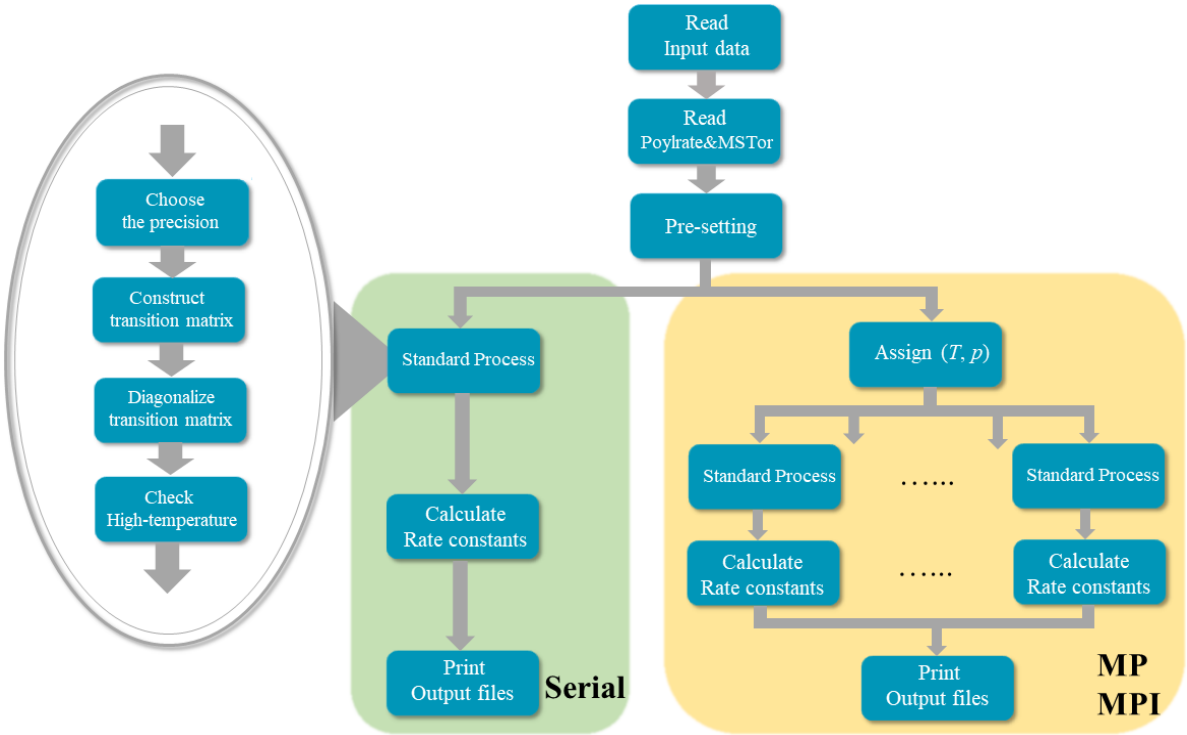


Figure 1. Workflow of the program

3.2 The Framework of Source Files

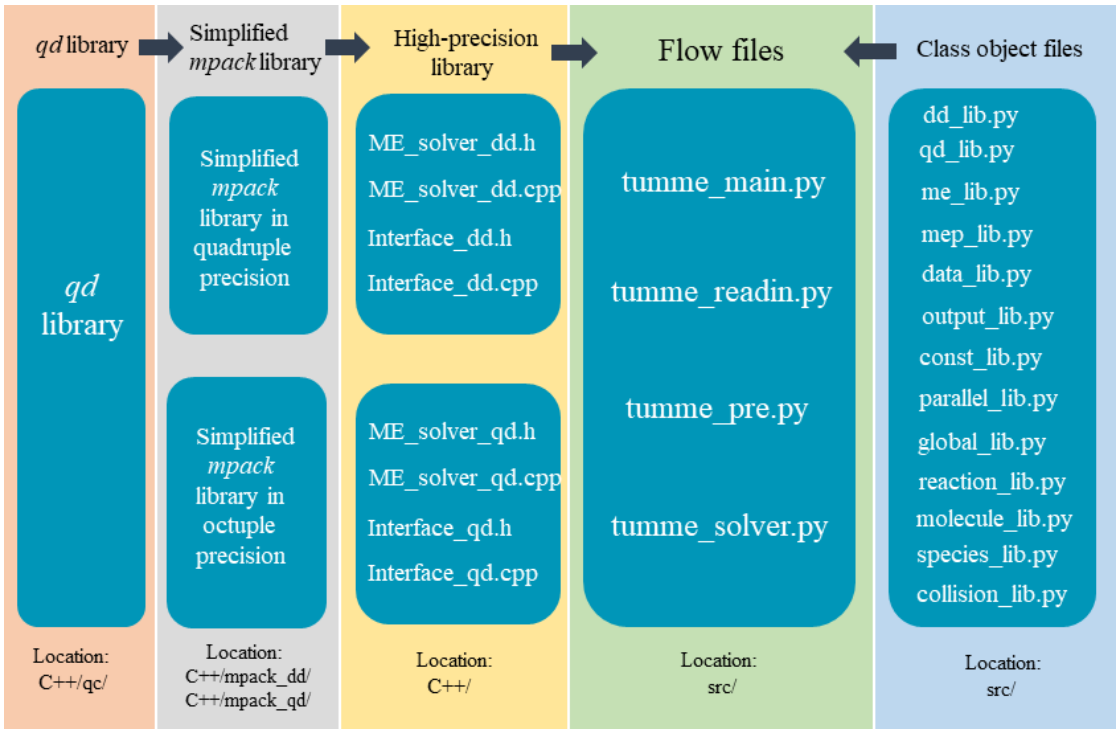


Figure 2. Framework of all source files

The main body of TUMME is written by Python 3 and is located in the src/ folder. Because the accurate calculation of eigenpairs of the transition matrix often requires greater than double precision, we also provide options for quadruple and octuple precision as implemented by C++. The high-precision library, based on the simplified multi-precision math library *mpack* and multi-precision float library *qd*, are compiled into dynamic libraries to let the Python process call them. Fig. 2 give a general picture showing how the source files work together. Section 5 of this manual (Detailed implementation) gives further information about each of the files.

3.3 Input Files

When the code is run, a standard input file which will be elaborated in section 4 will be read. *This standard input file is compulsory for any kind of job.* Besides, TUMME provides some features to let user use some advanced options. In those cases, some extra input files may be needed. For clarity, we listed here what kinds of options user can access and the required extra files for those options. The name of the standard input file is arbitrary; we recommend that it should end with the suffix ".in".

3.3.1 Options of Input Species Properties

In a master equation calculation, all species properties involved should be input for the program, including the coordinate geometries, frequencies, energies, symmetry number and so on. In TUMME, we provided three ways for user to input the required species information:

- Read from the **SPECIES** block in the standard input file of TUMME, in this case, only the standard input file is needed.
- Read from the standard output file of *Gaussian*[9], in this case, the standard output file of *Gaussian* is needed.
- Read from the standard output file of *Polyrate*[10], in this case, the standard output file of *Polyrate* is needed.

The priority for the program to read species properties is

Polyrate standard output file > *Gaussian* standard output file > **SPECIES** block

Frequencies of species calculated from diagonalizing the mass-scaled Hessian matrix should always be scaled a factor to correct the electronic potential energy surface error and the harmonic approximation error. Frequencies in *Polyrate* standard output is already scaled while in *Gaussian* standard output is unscaled. Thus, if frequencies for a species are read from a *Gaussian* standard output file, user should always set the subkeyword **FREQSCALE**. See section 4 for the detailed description.

3.3.2 Options of Microcanonical Flux Coefficients

3.3.2.1 Based on the framework of RRKM theory

The code can calculate the RRKM microcanonical flux coefficients under single-structural harmonic oscillator (SSHO) approximation (i.e., microcanonical conventional transition state theory flux coefficients in the harmonic approximation) without any other files (i.e., without any files beyond the standard input file). However, a key feature of TUMME is that – for reactions with an intrinsic barrier – the code can also calculate microcanonical flux coefficients that include anharmonicity, recrossing effects (i.e., variational effects), and tunneling. We list the available options for microcanonical flux coefficients in Table 1. This table shows that the program is capable of using any combination of these three effects.

We emphasize that TUMME does not read any microcanonical flux coefficients from external files; all flux coefficients are calculated internally. However, to go beyond microcanonical conventional transition state theory in the harmonic approximation, TUMME reads the necessary data from output files created by *Polyrate* and/or *MSTor*. These files must be available prior to the TUMME run; TUMME does not involve those packages, not does it run or spawn jobs involving those codes.

Note that whenever we refer to *Polyrate* output files, the meaning also includes files produced by *Polyrate* interfaces, for example, it could be a file produced by *Gaussrate*[11], which is an interface of *Polyrate* with *Gaussian*, or it could be a file produced by *NWChemRate*[12], which is an interface of *Polyrate* with *NWChem*.

It should be clearly pointed out that even though the keywords of these three effects are listed and set independently, effects per se are coupled with each other. E.g., the tunneling effect can have impact on the recrossing effect. Please refer to Ref.25 for details in theory.

TUMME only reads the following information from *Polyrate* output files:

- geometries, frequencies, electronic degeneracy, and energies of reactants,

conventional transition states, products, and all of generalized transition states along the minimum energy reaction path (MEP).

- zero-curvature transmission possibilities [$P^{ZCT}(E)$] and small-curvature transmission possibilities [$P^{SCT}(E)$]

The code will try to read all the above information. The portion of this information that will be used will depend on the specified keywords; other information is not used. The code will try to search particular key-strings to read the information (see Section 4.5.1); for example, if "*`transmission probability`*" is located in the file, the code will automatically find and read $P^{SCT}(E)$ and $P^{ZCT}(E)$ respectively.

Table 1. Features in Microcanonical Flux Coefficients

Effect	Method	Compulsory keyword	Compulsory extra files
Anharmonicity	SSHO[a]	Do not set **MSTFILE**	No extra file is needed.
	MS[b]	set **MSTFILE**	The standard output file of *MSTor*
Recrossing	TST[c]	set **VARIATION** as *tst*	No extra file is needed
	CVT[d]	set **VARIATION** as *cvt*	The standard output file of *Polyrate*
	μVT[e]	set **VARIATION** as *muvt*	The standard output file of *Polyrate*
Tunneling	SCT[f]	set **TUNNELING** as *sct*	The standard output file of *Polyrate*
	ZCT[g]	set **TUNNELING** as *zct*	The standard output file of *Polyrate*
	Eckart[h]	set **TUNNELING** as *Eckart*	No extra file is needed.
	No tunneling	Do not set **TUNNELING**	No extra file is needed.

[a] single-structure harmonic oscillator.[13] If frequencies are scaled before input, the approximation will be the quasiharmonic oscillator (QHO), which means that one uses the harmonic oscillator formulas with scaled or other effective frequencies. (Users should scale frequencies themselves prior to inputting them; TUMME has no keyword to scale the frequencies.) Note that in the formulas of Section 2.1.2, we use "HO" to denote either the harmonic oscillator approximation or the quasiharmonic oscillator approximation.

[b] multi-structural torsional approximation:[14] MS-TST[15] or MS-CVT[15] or MS-μVT[16]

[c] TST denotes conventional TST[13]

[d] a VTST option for canonical variational theory[17,18] or MS-CVT[15]. For each temperature, the program places the variational transition state at the point along the reaction path that maximizes Gibbs free energy of activation. This selected variational transition state will be used for the microcanonical flux coefficient for all energy bins.

[e] a VTST option for microcanonical variational theory[19,20] or MS-μVT[8]. For each energy bin, the program places the variational transition state at the point along the reaction path that minimizes the cumulative reaction probability or sum of states.

[f] zero-curvature tunneling[21,22,23]

[g] small-curvature tunneling[24]

[h] Eckart tunneling[25]

Here we explain the difference between the CVT and μVT options in TUMME for the variational effect. To begin we clarify two concepts, the generalized transition state and the variational transition state. The dividing surfaces through points along the MEP are generalized transition states. Only some of these are variational transition states. The actual structures on the MEP are generalized transition structures. Only some of these are variational transition structures. Generalized transition states and generalized transition structures that correspond to local maxima of the generalized free energy of activation for a given temperature are canonical variational transition states and canonical variational transition structures. Generalized transition states and generalized transition structures that correspond to a local minimum of the sum of states (or cumulative reaction probability) up to energy E are microcanonical variational transition states and microcanonical variational transition structures for that energy. Both canonical variational transition states and microcanonical variational transition states are called variational transition states. The difference between the CVT and μVT options is in the location of the variational transition state. If the standard output from *Polyrate* contains the information about the generalized transition states (geometries, frequencies, and energies are read; the electronic degeneracy and symmetry number are set equal to that of the conventional transition state), the code will read them and calculate the generalized Gibbs free energy of activation or the sum of vibrational-rotational states (or cumulative reaction probability) along the reaction path according to the CVT or μVT option. Under both options, TUMME computes microcanonical flux coefficients as functions of total energy. However, with the CVT option, it places the variational transition state at the location of the canonical variational transition state for the temperature in question, whereas with the μVT option it places it at the minimum of the sum over vibrational-rotational states (or cumulative reaction probability) for the total energy under consideration. The free energies of activation, the densities of vibrational-rotational states, and the sum of sum of vibrational-rotational states (or cumulative reaction probability) of the generalized transition states are calculated by quasiharmonic oscillator-rigid rotor approximation.

3.3.2.2 Based on the inverse-Laplace-transform to the high-pressure limits

For a specific elementary reaction, the high-pressure limit of rate constants (also called the canonical flux coefficient) equals the Laplace transform of the microcanonical flux coefficient, i.e., one can first get the canonical flux coefficients and then calculate the microcanonical one according to the inverse Laplace transform. In order to make TUMME more powerful and flexible, we also provide the inverse-Laplace-transform option for users. The inverse-Laplace transform option can be used by assigning **INVLAPLACE** subsection in the **BARRIERRXN** or **BARRIERLESSRXN** section.

4. Installation and Execution

4.1. Dependencies

Table 2. Environmental requirements of TUMME ***3.0***

Compulsory	>=Python 3.7.3 >=Numpy 1.16.2 >=Numba 0.43.1 >=Scipy 1.2.1
Optional	GNU compiler MPI Executor mpi4py module

The program was developed and debugged under the environment of *Anaconda3.2019.03* and we recommend that users install the latest *Anaconda* package. It should be possible to run serial and MP calculations in double-precision on a variety of platforms when the Python environment is set properly. We note though MPI calculations with high-precision libraries have only been developed and tested under Linux. Restated in terms of the above chart, we note that the compulsory part can be well run on various platform systems when the Python environment is well-set, and we note that the optional parts have only been developed and tested under Linux.

4.2. Installation

Table 2 gives the compulsory environments of the code. Provided that all compulsory environments set up properly, the program should run well in serial or MP mode in double precision. If the user wants to run it with MPI and/or with a quadruple-precision or octuple-precision library, one or both the following two steps should be carried out.

1) Install the python module *mpi4py*

To run it in MPI mode, make sure you have an MPI environment in your system; this should contain a *mpirun*-like command. Furthermore, you should install the *mpi4py* Python package. The installation of *mpi4py* is straightforward, and users can refer to the official website https://mpi4py.readthedocs.io/en/stable/install.html . If something goes wrong when you install the mpi4py, try to install the latest Anaconda beforehand.

2) Install the high-precision library

User can install the high-precision library by directly executing a script named *configure* located in the root path of TUMME as

```
./configure
```

4.2.1 Details in the *configure* script

The high-precision libraries can be run in serial, or they can be run in parallel with MP or MPI. To run with a quadruple-precision or octuple-precision library, you need to install some libraries in the C++/ folder. We provide a script called *configure* which is located at the root folder of TUMME to help users compile the libraries. Usually, it can be automatically installed.

If the script fails, the user can compile the libraries manually as follows.

① Set the root path of TUMME

```
path=`pwd`
```

② Install the *qd* library.

```
cd C++/qd/qd-2.3.7.1
./configure –prefix  $path/C++/qd CXX=g++ CC=gcc FC=gfortran
make
make install
```

③ Install the *mpack_dd* libraries.

```
cd $path/C++/mpack_dd/src
make
```

④ Install the *mpack_qd* libraries.

```
cd $path/C++/mpack_qd/src
make
```

⑤ Install the high-precision library.

```
cd $path/C++/
 make
# Users should modify the variable $ROOTPATH in Makefile of the high-
# precision library to be the absolute root path of the program. For
# example, if the root path is /app/TUMME 3.0, then user should find
# the makefile /app/TUMME_v3.0/C++/Makefile, and replace the sixth line
#                   ROOTPATH =[replace this line]
# by the following line
#                   ROOTPATH =/app/TUMME 3.0
```

⑥ Set environment variable for TUMME

```
echo "# environment variable of TUMME" >> ~/.bashrc
echo 'export PATH=$PATH:'$path'/bin' >> ~/.bashrc
```

The library now works properly with the GNU compiler, but not with the Intel compiler. So, the user should make sure to use GNU compiler commands: *gcc*, *g++*, and *gfortran*. We have tested the compatibility of the high-precision library code with the 4.8.5, 4.9.2, 5.1.0, 5.4.0, 6.1.0, 6.3.0, 7.2.0, 8.1.0, 8.2.0 and 9.2.0 versions of the GNU compiler. Other versions have not been tested but should also work well.

The folder *example/2MH/highprecision/* contains benchmark outputs of the eigenvector in quadruple and octuple precision. To validate the high-precision library, the user can access this folder and run TUMME including print out of the CSE eigenvector (by setting **EVECNUM** as *1*) in quadruple and octuple precision (by setting **PRECISION** as *quadruple* or *octuple*). Since this example is a single-well dissociation reaction, all values in the CSE eigenvector should have the same sign, and the absolute value should decrease as the energy bin increases. (In this manual, well and isomer are synonyms.) The value of the eigenvector in high-energy bins can reflect the precision of the floating number you used. When the value in high-energy bins oscillates between negative and positive signs, it is an indication that one needs higher precision. In octuple precision, over the energy bins in the range 1–77 kcal/mol, the eigenvector values should have the same sign and the value should be of the order of magnitude of 10^{-64} in the ~77 kcal/mol energy bin. In quadruple precision, in energy bins over the approximate range 1~54 kcal/mol, the eigenvector values should have

the same sign and the value of energy bin at ~ 54 kcal/mol should be of the order of magnitude of 10^{-32}. If you used the high-precision option, but the values of high-energy bins oscillate at a magnitude of about 10^{-16}, your high-precision library is not properly installed, and you should reinstall it.

4.3. Execution

We provide an executable script called *tumme* to run the program located in the folder of bin/, so user can easily run TUMME in any directory as

```
tumme param.in
```

4.3.1 Details in the executable script *tumme*

1) *Read parallelism mode and number of processors from the standard input file.* Users should specify the "**#PARALLEL**" and "**#NPROC**" values in the standard input file if a parallel scheme is adopted (refer to Section 5.1). The parallel scheme is designed for running multiple temperatures and multiple pressures. The set of (T, p) will be divided evenly and each processor will run its own subset. So, if you only run one temperature and one pressure, you should not use a parallel scheme.

2) *Execute python according to the parallelism.* The general form of command to execute the program is:

```
$mpi $python3 $path/tumme_main.py $input
```

where `$python3` is the command for Python 3; `$path` is the absolute path for the folder src/ of TUMME; and `$input` is the name of the standard input file.

For the serial and MP runs, `$mpi` is a null string or spaces, e.g.

```
python3 /home/tumme/src/tumme_main.py param.in
```

For the MPI runs, `$mpi` is '`mpirun -np 4`'-like string, e.g.

```
mpirun -np 4 python3 /home/tumme/src/tumme_main.py param.in
```

4.3.2 The parallelism

The parallelism of the high-precision libraries in TUMME requires further clarification. The high-precision libraries are implemented in C++. In order to decrease the execution time, we utilized OpenMP to parallelize it. This use of OpenMP is independent of whether the program is run with the serial, MP, or MPI scheme in Python. MP and MPI are used for parallelizing the Python code, while OpenMP is used for parallelizing the C++ code. When the code is run in serial, MP, or MPI mode, the system will create *nproc* threads or processes running the Python code. Each thread or process will calculate its own subset of $\{(T, p)\}$. If the high-precision option is selected, each thread or process will call the C++ dynamic library and will further be parallelized into m threads by OpenMP. The user can control the value of m by setting the subkeyword **OMPNUMTHREAD**. If not set, the program will by default set m to be the total number of processors a node owns divided by **NPROC**. The total number of processors of a node will be obtained by Python using the *os.cpu_count()* command. This dynamic default value will maximize the efficiency of a single-node calculation but may be inappropriate for multi-node computations because **NPROC** will not be the number of Python threads/processes running on a node. The user should change **OMPNUMTHREAD** according to the following rule:

the total number of OpenMP threads created by the Python threads or processes on a node should not exceed the total number of processors on that node.

5. Standard Input File

The input file is divided into blocks. Each block has sections, and some sections have subsections. All blocks, sections, and subsections have a start-keyword and an end-keyword. All blocks, sections, and subsections can have keywords that define values; we call this kind of keyword a subkeyword. In this text, keywords and subkeywords are in bold Arial font.

There are three kinds of subkeywords: string subkeywords, value subkeywords, and list subkeywords. All the start-keywords and list subkeywords have an end-keyword. An end-keyword is equal to the corresponding start-keyword or list subkeyword with an added prefix **END_**, e.g., for a start-keyword **PARAMETER**, the end-keyword is **END_PARAMETER**. The end-keywords and end-subkeywords are not shown in Table 3, but they are shown in Tables 4–6.

Only subkeywords can have a value, e.g., string, float, or list of floats. (Throughout the manual we shorten "floating point number" to "float".) String values should not contain spaces. Floating point number subkeywords can accept formats like "1.234", ".234", "1.", ".23E2", "1.23E1"; it is not acceptable to replace "E" with "D". Values and subkeywords are separated by a space. Subkeywords can have a suffix string to tell the program the unit of the values, e.g., **PRESSURE**[torr]. The available suffix strings for units are:

energies	[kcal/mol]
	[kJ/mol]
	[a.u.]
	[eV]
	[cm-1]
frequency	[cm-1]
	[a.u.]
pressures	[bar]
	[atm]
	[torr]
	[a.u.]
distances	[A]
	[bohr]
temperature	[K]
	[a.u.]
mass	[amu]
	[a.u.]
time	[s]
	[ms]
	[ps]
	[fs]
	[a.u.]
rate constant	[cm3/molecule/s]
	[1/s]

The brackets are required. If the unit is not specified, the unit will default to atomic units.

All start-keywords, end-keywords, and subkeywords are case insensitive. We use upper case in the manual, but in the actual input file, they can be lower case or mixed case. But letters in a unit string and a value are case sensitive, so the user should use precisely the cases shown in this manual.

The standard input file has two pre-definition subkeywords, **#PARALLEL** and **#NPROC**, and three blocks: **PARAMETER** block, **REACTION** block, and **SPECIES** block.

#PARALLEL and **#NPROC** define the parallel mode and specify the number of processors. If the two pre-definition subkeywords do not appear, then serial mode is adopted.

The **PARAMETER** block sets all parameters needed for solving the master equation except for the elementary reactions and species properties.

The **REACTION** block specifies the properties of the elementary reactions.

The **SPECIES** block defines all species properties.

The **PARAMETER** block should appear before the **REACTION** block. The **REACTION** block should appear before the **SPECIES** block. The **PARAMETER** block and **REACTION** block are compulsory, but the **SPECIES** block is optional.

The user has three ways to provide species properties to the program: from a *Polyrate* output file, from a *Gaussian* output file, and/or from the **SPECIES** block. The priority for the program to read species properties is

Polyrate output file > *Gaussian* output file > **SPECIES** block

In the **REACTION** block, if **PYRFILE** (a subkeyword to specify the name of the *Polyrate* output file) is defined for a **BARRIERRXN** section (a subkeyword to specify an elementary reaction that has a barrier), the species will be read from the *Polyrate* output file and be named according to the **INFO** string. When a species is read from a *Polyrate* output file, the species do not have to be defined in the **SPECIES** block. If it were redundantly defined anyway in the **SPECIES** block, all properties will be ignored except **ROTSIGMA**, **OPTICALNUM,** and **MSTFILE,** which are the three subkeywords for the rotational symmetry number, the optical-isomer number, and the name of *MSTor* output file. In the **SPECIES** block, if **G09FILE** is defined in a species section, then geometries, frequencies, symbols, rotational symmetry numbers, the imaginary frequency and energies will be read from the *Gaussian* output file; other properties e.g., name string, optical-isomer number, and *MSTor* file name string, will be read from the **SPECIES** block. If some species are read from *Gaussian* output files, the user should make sure that the zero of energy is consistent with those read from the **SPECIES** block.

Except **#PARALLEL** and **#NPROC**, any line started with # will be treated as a comment and skipped. Blank lines are acceptable in any place.

The general schematic of all keywords is presented in Table 3.

Table 3. Keywords in the standard input file (excluding end-keywords)

Keyword	Keyword type	Default
#PARALLEL	String subkeyword	*serial*
#NPROC	Value subkeyword	1
PARAMETER	Start-keyword of block	--

LJCOLLISION	Start-keyword of section	--
DIAMETERM	Value subkeyword	None
DIAMETERA	Value subkeyword	None
EPSILONM	Value subkeyword	None
EPSILONA	Value subkeyword	None
MASSM	Value subkeyword	None
MASSA	Value subkeyword	None
HSCOLLISION	Start-keyword of section	--
BMAX	Value subkeyword	None
MASSM	Value subkeyword	None
MASSA	Value subkeyword	None
ENERGY	Start-keyword of section	--
DE	Value subkeyword	1.0 cm^{-1}
EMAX	Value subkeyword	400 kcal/mol
ESOT	Value subkeyword	0.1
EEOT	Value subkeyword	30
PBIMOL	Start-keyword of section	--
BIMOLNAME	List subkeywords	--
EXCESSCONC	List subkeywords	--
PRINT	Start-keyword of section	--
EIGENVALUE	Start-keyword of subsection	--
EVALFILE	String subkeyword	None
EVALNUM	Value subkeyword	None
EIGENVECTOR	Start-keyword of subsection	--
EVECFILE	String subkeyword	None
EVECNUM	Value subkeyword	None
PARTITIONFUNCTION	String subkeyword	None
MICROFLUXCOEFFICIENT	String subkeyword	None
TIMEEVOLUTION	Start-keyword of subsection	--
TIMEFILE	String subkeyword	None
TIMESTEP	Value subkeyword	$0.02/\lambda_0$
TIMENUM	Value subkeyword	50
INITISPECIES	List subkeyword	None
RELAXKERNEL	Start-keyword of subsection	--
RELAXFILE	Value subkeyword	None
KERNELNAME	List subkeyword	None
MERGETHRESHOLD	Value subkeyword	0.2
PRECISION	String subkeyword	*double*
OMPNUMTHREAD	Value subkeyword	*cpu_count /nproc*
EDOWN	Start-keyword of subsection	--
SLOPE	Value subkeyword	0
EDCONST	Value subkeyword	None
TC	Value subkeyword	300 K
EXPONENT	Value subkeyword	0
TEMPERATURE	List subkeyword	None
PRESSURE	List subkeyword	None
GROUNDSPECIES	String subkeyword	None
REACTION	Start-keyword of block	--
BARRIERRXN	Start-keyword of section	--
INFO	String subkeyword	None
PYRFILE	String subkeyword	None
VARITAION	String subkeyword	None
TUNNELING	String subkeyword	None
BARRIERLESSRXN	Start-keyword of section	--
INFO	String subkeyword	None

RXNENERGY	Value subkeyword	None
AVGDIAMETER	Value subkeyword	None
SPECIES	Start-keyword of block	--
WELL	Start-keyword of section	--
EPSILON	Value subkeyword	None
BMAX	Value subkeyword	None
DIAMETER	Value subkeyword	None
TRANSTATE	Start-keyword of section	--
IMAGFREQ	Value subkeyword	None
BIM	Start-keyword of section	--
NAME	String subkeyword	None
E0	Value subkeyword	None
EELE	Value subkeyword	None
SP1MOL	Start-keyword of subsection	--
SP2MOL	Start-keyword of subsection	--
SP1ATOM	Start-keyword of subsection	--
SP2ATOM	Start-keyword of subsection	--
Subkeywords for molecular species [a]		
NAME	String subkeyword	None
GEOMETRY	List subkeyword	None
FREQUENCY	List subkeyword	None
ELELEVEL	List subkeyword	1 0
E0K	Value subkeyword	None
EELE	Value subkeyword	None
ROTSIGMA	Value subkeyword	1
OPTICALNUM	Value subkeyword	1
FREQSCALE	Value subkeyword	1.0
G09FILE	String subkeyword	None
MSTFILE	String subkeyword	None
Subkeywords for atomic species [b]		
NAME	String subkeyword	None
SYMBOL	String subkeyword	None
ELELEVEL	List subkeyword	1 0
E0K	Value subkeyword	None
EELE	Value subkeyword	None
Subkeywords for inverse Laplace transform [c]		
INVLAPLACE	Start-keyword of subsection	--
EXPRESSION	String subkeyword	--
PREFACTOR1	Value subkeyword	--
T1	Value subkeyword	--
N1	Value subkeyword	--
EA2	Value subkeyword	--
PREFACTOR2	Value subkeyword	--
T2	Value subkeyword	--
N2	Value subkeyword	--
EA2	Value subkeyword	--

a. These subkeywords are available for the molecular species sections/subsections: **WELL**, **TRANSTATE**, **SP1MOL**, and **SP2MOL**.

b. These subkeywords are available for the atomic species subsections: **SP1ATOM** and **SP2ATOM**.

c. The inverse Laplace transform subsection **INVLAPLACE** could be used in both **BARRERRXN** and **BARRIERLESSRXN** sections.

5.1 Two pre-definition subkeywords

To run the program in MP or MPI mode, the user should specify the value **#PARALLEL** in the first line of the input file. **#PARALLEL** can be *serial*, *mp*, or *mpi*. When *mp* is specified, **#NPROC** should be specified as the second line to determine how many processors to use; but for *mpi* the number of processors is determined by the `$mpirun` command, and **#NPROC** is not needed for the *mpi* mode.

Example:

#PARALLEL *mp*

#NPROC *4*

See Section 4.3 for more discussion of parallelism.

5.2 Keywords in the PARAMETER block

Table 4. Glossary of the PARAMETER block

PARAMETER	
Description:	Start-keyword for a block describing all global parameters. This block is compulsory in the standard input file.
Available sections and subkeywords:	**LJCOLLISION**, **HSCOLLISION**, **ENERGY**, **PRINT**, **MERGETHRESHOLD, PRECISION, EDOWN, TEMPERATURE, PRESSURE, GROUNDSPECIES**
Example:	**PARAMETER** **LJCOLLISION** **……** **END_LJCOLLISION** **……** **END_PARAMETER**
LJCOLLISION	
Description:	Start-keyword for the Lennard-Jones collision section. Here we present two kinds of collision models, one for the Lennard-Jones collision model and the other for the hard-sphere collision model. The former is set by **LJCOLLISION** section, and the latter is set by **HSCOLLISION**. Users should define only one of them. The equation used to estimate the Lennard–Jones collision rate constant is $$k_{\mathrm{LJ}} = \Omega^*_{2,2}\,\pi\left(\frac{d_\gamma + d_{\mathrm{M}}}{2}\right)^2\sqrt{\frac{8k_{\mathrm{B}}T}{\pi m_\gamma m_{\mathrm{M}}/(m_\gamma + m_{\mathrm{M}})}}$$ $$\Omega^*_{2,2} = \begin{cases}\left[0.636 + 0.567\log_{10}\left(\frac{k_{\mathrm{B}}T}{\sqrt{\varepsilon_\gamma\varepsilon_{\mathrm{M}}}}\right)\right]^{-1}, & \frac{k_{\mathrm{B}}T}{\sqrt{\varepsilon_\gamma\varepsilon_{\mathrm{M}}}} \in [0.3,3] \\ \left[0.697 + 0.5185\log_{10}\left(\frac{k_{\mathrm{B}}T}{\sqrt{\varepsilon_\gamma\varepsilon_{\mathrm{M}}}}\right)\right]^{-1}, & \frac{k_{\mathrm{B}}T}{\sqrt{\varepsilon_\gamma\varepsilon_{\mathrm{M}}}} \in [3,300]\end{cases}$$ where γ denotes an isomer; M denotes the bath gas; d denotes the diameter; ε is the Lennard-Jones energy parameter; and m denotes the mass. If the argument is not in the range from 0.3 to 300, the code will print a warning. The reference for these approximations to $\Omega^*_{2,2}$ is a 1977 paper by Troe.[26]
Available subkeywords:	**DIAMETERM**, **DIAMETERA**, **EPSILONM**, **EPSILONA**, **MASSM**, **MASSA**
Example:	**LJCOLLISION** **DIAMETERM**[A] *3.0* **DIAMETERA**[A] *4.0* **EPSILONM**[cm-1] *3.0* **EPSILONA**[cm-1] *3.0* **MASSM**[amu] *40* **MASSA**[amu] *90* **END_LJCOLLISION**

DIAMETERM

Description:	Subkeyword for the diameter d_{M} of a molecule of the bath gas. This is compulsory in the **LJCOLLISION** section.
Available values:	Positive float
Example:	**DIAMETERM**[A] *3.0*

DIAMETERA

Description:	Subkeyword for the general diameter d_{γ} of the isomers. (In this manual, well and isomer are synonyms.) If specified, all isomers will have the same diameter, **DIAMETERA**. If not, the diameter is isomer-specific, and in that case the **DIAMETER** subkeyword should be assigned in all **WELL** sections.
Available values:	Positive float
Example:	**DIAMETERA**[A] *3.0*

MASSM

Description:	Subkeyword for the mass m_{M} of the bath gas molecule. This is compulsory in the **LJCOLLISION** section.
Available values:	Positive Float
Example:	**MASSM**[amu] *40.0*

MASSA

Description:	Subkeyword for the general mass m_{γ} of the isomers. (In this manual, well and isomer are synonyms.) The masses of all **WELL** species must be the same. The user can specify this mass as **MASSA**; however, if **MASSA** is not specified the program will calculate the mass from **GEOMETRY** data in the first **WELL** section.
Available values:	Positive float
Example:	**MASSA**[amu] *50.0*

EPSILONM

Description:	Subkeyword for the energy parameter ε_{M} of the bath gas molecule. This is compulsory in the **LJCOLLISION** section.
Available values:	Positive float
Example:	**EPSILONM**[cm-1] *3.0*

EPSILONA

Description:	Subkeyword for the general energy parameter ε_γ of the isomers. If specified, all isomers will have the same Lennard-Jones energy parameter, which will be equal to **EPSILONA**. If not, the energy parameter is isomer-specific, and in that case the **EPSILON** subkeyword should be assigned in all **WELL** sections. (In this manual, well and isomer are synonyms.)
Available values:	Positive float
Example:	**EPSILONA**[cm-1] *3.0*

HSCOLLISION

Description:	Start-keyword for hard-sphere direct-dynamics collision section. Note that this keyword relates to collisions between the bath gas and isomers (it is not related to collisions between two members of a bimolecular pair undergoing a barrierless reaction). This keyword is to be used if the user has used direct dynamics to calculate the relaxation parameter $\langle \Delta E_{\mathrm{down}} \rangle$. If this option is chosen, user should input the maximum impact parameter b_{max} that was used in calculating $\langle \Delta E_{\mathrm{down}} \rangle$, and the collision rate constant will be calculated by $$k_{\mathrm{HS}} = \pi (b_{\mathrm{max}})^2 \sqrt{\frac{8 k_{\mathrm{B}} T}{\pi m_\gamma m_{\mathrm{M}} / (m_\gamma + m_{\mathrm{M}})}}$$
Available subkeywords:	**BMAX**, **MASSM**, **MASSA**

Example:	**HSCOLLISION**
	  BMAX[A] *9.0*
	  MASSM[amu] *40*
	  MASSA[amu] *90*
	END_HSCOLLISION

BMAX

Description:	Subkeyword to be used when the same maximum impact parameter b_{max} is to be assigned to all isomers. If this subkeyword is not specified in **HSCOLLISION** section, **BMAX** should be specified in each **WELL** section. (In this manual, well and isomer are synonyms.)
Available values:	Positive float
Example:	**BMAX**[A] *3.0*

ENERGY

Description:	Start-keyword for **ENERGY** section. This section is compulsory in the **PARAMETER** block.
Available subkeywords:	**DE**, **EMAX**, **ESOT**, **EEOT**

Example:

ENERGY
DE[cm-1] *1*
EMAX[kcal/mol] *40*
ESOT *0.1*
EEOT *30*
END_ENERGY

Background: The internal energy of each well (i.e., each isomer) is discretized into bins. The energy of bin η is

$$E_\eta = E_{\max} - (\eta - 1)\Delta E, \qquad \eta = 1, 2, \ldots, N_\gamma$$

DE

Description: Subkeyword for the energy step dE used in calculating the density of states by the Beyer–Swinehart algorithm. The default value is 1 cm^{-1}.

DE should be distinguished from the energy step set by **ESOT**. **DE** sets the energy step for calculating the density of states in the Beyer–Swinehart algorithm, whereas **ESOT** sets the energy step for the transition matrix. Usually, **DE** should be much smaller than the energy step set by **ESOT**.

Available values: Positive float

Example: **DE**[cm-1] *1.0*

EMAX

Description: Subkeyword for the maximum energy relative to the enthalpy at 0 K of each isomer used in calculating the density of states by the Beyer–Swinehart algorithm. The default value is 400 kcal/mol. **EMAX** should be distinguished from the maximum energy set by **EEOT**. **EMAX** sets the energy upper boundary for calculating the density of states in the Beyer–Swinehart algorithm, whereas **EEOT** sets the energy upper boundary for the transition matrix. Usually, **EMAX** should be much greater than the maximum energy set by **EEOT**.

Available values: Positive float

Example: **EMAX**[kcal/mol] *400*

ESOT

Description: Subkeyword for the ratio of energy bin width ΔE divided by $k_B T$. Based on this subkeyword, the energy step between energy bins in the master equation is set as follows,

$$\Delta E = ESOT \times k_B T$$

The default value is 0.1.

When the temperature is low, molecules tend to populate at low energy levels, so user should decrease **ESOT** to make the result converged.

Available values: Positive float

Example: **ESOT** *0.1*

EEOT

Description: Subkeyword used to set the maximum energy relative to the highest E_0^{sp} used in the transition matrix for solving the master equation. Note that E_0^{sp} is the energy at 0 K of reactants, transition states and products, where we use 0 K to specify the inclusion of ZPE but no thermal effects (ZPE denotes vibrational zero-point energy). The maximum energy in the master equation equals

$$\text{Maximum energy} = \max\{E_0^{\mathrm{sp}}\} + EEOT \times k_{\mathrm{B}}T$$

When **ESOT** and **EEOT** are determined, the number of energy bins for γ-th isomer is calculated as

$$N_\gamma = \mathrm{int}\left(\frac{\max\{E_0^{\mathrm{sp}}\} + EEOT \times k_{\mathrm{B}}T - E_{0,\gamma}}{\Delta E}\right)$$

where int denotes rounding down to an integer; ΔE is the energy step for each energy bin described in **ESOT**; and $E_{0,\gamma}$ is the ground-state energy (including vibrational zero-point energy) of the γ-th isomer. In this manual, well and isomer are synonyms.

When the temperature is high, molecules tend to populate at high energy levels, so user should increase **EEOT** to make the result converged.

The default value is 30.

Available values: Positive float

Example: **EEOT** *30*

PBIMOL

Description: The start-keyword of the subsection for the pseudo-first-order bimolecular pairs. In order to deal with the inhomogeneous term caused by a bimolecular (second-order) reaction, we treat bimolecular products as sinks, and we calculate bimolecular rate constants in the limit of pseudo-first-order kinetics in which one member of the reactant bimolecular pair is present in great excess so that its concentration is taken as a constant). The present subsection defines the bimolecular reactants and specifies the constant concentration of the excessive fragment. We stress here that in TUMME we assume that the pressure is caused by the bath gas, and the partial pressure of reagents is neglected in the collisional relaxation process. For a bimolecular reaction of A with B, with B present in great excess and M being the bath gas (present in even greater excess, we require $[\mathrm{M}] \gg [\mathrm{B}] \gg [\mathrm{A}]$.

Available subkeywords: **BIMOLNAME, EXCESSCONC**

Example:

PBIMOL
  BIMOLNAME
    Bim1 Bim2 Bim3
  END_ BIMOLNAME
  EXCESSCONC[mol/L]
    1E-5 1E-4 1E-3
  END_ EXCESSCONC

END_ PBIMOL

BIMOLNAME

Description: Subkeyword for the name-list of pseudo-first-order bimolecular reactants. User can assign more than one bimolecular pair. Name strings should be split with spaces.

Example:
BIMOLNAME
Bim1 Bim2 Bim3
END_ BIMOLNAME

EXCESSCONC

Description: Subkeyword for the constant concentration of the excessive reagent of pseudo-first-order bimolecular reactants. Since there could be more than one pseudo-first-order bimolecular reaction, there could be more than one value for this subkeyword. Values should be entered as floating-point numbers separated by spaces. The number of values should be the same as the number of pseudo-first-order bimolecular reactants, also the name string in **BIMOLNAME.**

Example:
EXCESSCONC[mol/L]
1E-5 1E-4 1E-3
END_ EXCESSCONC

PRINT

Description: Start-keyword for an output section.

Available subsections and subkeywords: **EIGENVALUE, EIGENVECTOR, PARTITIONFUNCTION MICROFLUXCOEFFICIENT, TIMEEVOLUTION**

Example:
PRINT
EIGENVECTOR
EVECFILE *evec.txt*
EVECNUM *3*
END_EIGENVECTOR
PARTITIONFUNCTION *Q.txt*
END_PRINT

EIGENVALUE

Description: Start-keyword for the eigenvalues output subsection.

Available subkeywords: **EVALFILE**, **EVALNUM**

Example:
EIGENVALUE
EVALFILE *eval.txt*
EVALNUM *3*
END_EIGENVALUE

EVALFILE

Description: Subkeyword for the file name of eigenvalues output. This is compulsory in the **EIGENVALUE** subsection. String value should not contain any space.

Available value: String

Example: **EVALFILE** *eval.txt*

EVALNUM

Description: Subkeyword for the number of eigenvalues to be printed. All eigenvalues are sorted in ascending order, and the first *evalnum* eigenvalues will be printed. If this value is not defined in the **EIGENVALUE** subsection, the program will print out all CSE eigenvalues and the minimum and the maximum IERE eigenvalues.

Available value: Positive integer

Example: **EVALNUM** *3*

EIGENVECTOR

Description: Start-keyword for the eigenvectors output subsection.

Available subkeywords: **EVECFILE**, **EVECNUM**

Example:

```
EIGENVECTOR
   EVECFILE  evec.txt
   EVECNUM  3
END_EIGENVECTOR
```

EVECFILE

Description: Subkeyword for the file name of eigenvectors output. String value should not contain any space.

Available value: String

Example: **EVECFILE** *evec.txt*

EVECNUM

Description: Subkeyword for the number of eigenvectors to be printed. All eigenvectors are sorted according to corresponding eigenvalues' sequence, and the first *evecnum* eigenvectors will be print. If this value is not defined in the **EIGENVECTOR** subsection, the program will print out all CSE eigenvectors.

Available value: Positive integer

Example: **EVECNUM** *3*

PARTITIONFUNCTION

Description: Subkeyword for the file name of partition function output. The partition function of all species in **TEMPERATURE** will be printed. If **MSTFILE** is not specified for a species, its partition function will be calculated by the single-structural rigid rotor and harmonic oscillator (SSHO) approximation. If **MSTFILE** is specified, the partition function of this species will be calculated by doing the Laplace transform of the read density of states(DoS). The electronic partition function, vibrational partition function, rotational partition function, and translational partition function of all isomers, transition states, members of bimolecular pairs will be printed. When the generalized transition states are read from **PYRFILE,** only the partition function of GTS estimated by the SSHO

approximation is printed. The program makes the approximation that

$$Q^{\mathrm{MST}}(s) \approx \frac{Q^{\mathrm{MST}}(s=0)}{Q^{\mathrm{SSHO}}(s=0)} Q^{\mathrm{SSHO}}(s)$$

and only prints $Q^{\mathrm{SSHO}}(s)$.

Available value: String

Example: **PARTITIONFUNCTION** *Q.txt*

MICROFLUXCOEFFICIENT

Description: Start-keyword for the file name of microcanonical flux coefficient output. All microcanonical flux coefficients used to construct a transition matrix will be printed. For the reaction from an isomer γ to an isomer/bimolecular ϕ, the microcanonical flux coefficient that is printed is

$$\hat{k}^{\mathrm{MS-VTST/SCT}}(\gamma \to \phi \mid E_\eta) = \kappa^{\mathrm{SCT}}(E_\eta)\Gamma^{\mathrm{VTST}}(E_\eta)F^{\mathrm{MS}}(E_\eta)\frac{N_{\gamma,\phi}^{\ddagger,\mathrm{SSHO}}(E_\eta)}{h\rho_\gamma^{\mathrm{SSHO}}(E_\eta)}$$

For the reaction from a bimolecular pair ν to an isomer γ, the flux coefficient that is

$$\Delta\hat{k}^{\mathrm{MS-VTST/SCT}}(\nu \to \gamma \mid E_\eta) = \Gamma^{\mathrm{VTST}}(E_\eta)\kappa^{\mathrm{SCT}}(E_\eta)F^{\mathrm{MS}}(E_\eta)\frac{N_{\nu,\gamma}^{\ddagger,\mathrm{SSHO}}(E_\eta)e^{-\beta E_\eta}\Delta E}{h\Phi_{\mathrm{rel}}Q_\nu^{\mathrm{SSHO}}}$$

where Γ, κ and F are respectively the microcanonical recrossing transmission coefficient, the tunneling transmission coefficient, and the multiple-structure torsional coefficient, and they are dependent on the settings of **VARIATION**, **TUNNELING**, and **MSTFILE** subkeywords in a coupled way; N is the sum of states, Q is the electronic-vibrational-rotational partition function, and Φ is the relative translation partition function per unit volume. A more detailed description is given in Ref. 25.

Available values: String

Example: **MICROFLUXCOEFFICIENT** *kE.txt*

TIMEEVOLUTION

Description: Start-keyword of the subsection for the time evolution of populations. In TUMME, the initial conditions (populations at time zero) are set as a kronecker-delta condition. This means that the population of the ground-state energy bin of the reactant of the first reaction is 1, and the populations of all of the other energy bins of this reactant and all of the energy bins of other species are set to zero. The equations for the time evolution are presented in the *Faraday Discussions* paper.[27]

Available subkeywords: **TIMEFILE**, **TIMESTEP**, **TIMEMAX**, **INITSPECIES**

Example:

TIMEEVOLUTION
  TIMEFILE *time.txt*
  TIMENUM *100*
  INITSPECIES *Well_1*
END_ TIMEEVOLUTION

TIMEFILE

Description: Subkeyword for the output file name for the time evolution of populations. This is compulsory in the **TIMEEVOLUTION** subsection.

Available values: String

Example: **TIMEFILE** *time.txt*

TIMESTEP

Description: Subkeyword for the time step of time evolution of populations to be printed. The default value is

$$\Delta t = \frac{1}{50\lambda_0}$$

where λ_0 is the smallest non-zero eigenvalue.

Available values: Positive float

Example: **TIMESTEP**[fs] *0.1*

TIMENUM

Description: Subkeyword for the number of steps of time (N_{time}) evolution output. The first N_{time} steps will be printed. The default value is

$$N_{\text{time}} = 50$$

Available values: Positive integer

Example: **TIMENUM** *100*

INITSPECIES

Description: Subkeyword for names or names of the reactants of the first reaction. It can be either a bimolecular pair or a well. The bimolecular pair should be defined in the **PSEUBIMOLECULAR**. This subkeyword is compulsory in the **TIMEEVOLUTION** subsection.

Available values: String List

Example:

INITSPECIES
Well_1
Bim_2
END_INITSPECIES

RELAXKERNEL

Description: Start-keyword of the subsection for relaxation kernels $P(E|E')$.

Available subsections and subkeywords: **RELAXFILE, KERNELNAME**

Example:

RELAXKERNEL
RELAXFILE *kernel.out*
KERNELNAME
Well_1 Well_2 Well_3
END_KERNELNAME
END_RELAXKERNEL

RELAXFILE

Description: Subkeyword for a file name of the output of relaxation kernels. This subkeyword is compulsory in the **RELAXKERNEL** subsection. There is no default; the user must provide it.

Available values: String

Example: **RELAXFILE** *kernel.out*

KERNELNAME

Description: Subkeyword for the list of isomer names of which the relaxation kernel will be printed. Users can input multiple isomer names which should be separated by spaces or by a line break. This subkeyword is compulsory in the **RELAXKERNEL** subsection.

Available values: String list

Example: **KERNELNAME**

Well_1

Well_2 Well_3

END_KERNELNAME

MERGETHRESHOLD

Description: Subkeyword for the threshold to determine if a merge occurred. The **MERGETHRESHOLD** is compared to the squared projection of eigenvectors onto the IERE subspace (refer to Section 5). The default value is 0.2.

Available values: Positive float

Example: **MERGETHRESHOLD** *0.2*

PRECISION

Description: Subkeyword for the precision of the transition matrix and the eigenpairs. The default value is *double*. Double precision based on the float64 data type will provide ~16 decimal places; quadruple precision based on the double-double data type will provide ~32 decimal places; and octuple-precision based on the quadruple-double data type will provide ~64 decimal places.

Available values: *double, quadruple, octuple*

Example: **PRECISION** *double*

OMPNUMTHREAD

Description: Subkeyword for the number of threads set in OpenMP. The high-precision libraries are implemented by C++ where OpenMP is utilized in order to decrease the high-precision time consumption. We stress that the OpenMP parallelization is separate from the MP and MPI parallelization. The **OMPNUMTHREAD** subkeyword controls into how many threads each Python thread/process will be further parallelized, and it is equivalent to the system environment variable `OMP_NUM_THREADS`. This subkeyword only works when

the **PRECESION** subkeyword is set to be *quadruple* or *octuple*. If this subkeyword is not set, the program will by default set it to be the total number of processors a node owns divided by the **NPROC** value. The total number of processors will be obtained by Python using *os.cpu_count()*. This dynamic default value will maximize the efficiency of single-node calculations but may be inappropriate for multi-nodes computation because **NPROC** will not be the number of Python threads/processes run on a node, and the user should change this value accordingly.

Available values: Positive integer.

Example: **OMPNUMTHREAD** *2*

EDOWN

Description: Start-keyword of the subkeyword for the average downwards energy transfer $\langle \Delta E_{\mathrm{d}} \rangle$. This is used in the exponential down relaxation kernel. See Eq. (11) for the definition $\langle \Delta E_{\mathrm{d}} \rangle$. The $\langle \Delta E_{\mathrm{d}} \rangle$ moment is treated as having the following dependence on the temperature of the bath and on the initial internal energy of the isomer:

$$\langle \Delta E_{\mathrm{d}} \rangle = \left(\frac{T}{T_{\mathrm{c}}}\right)^{n} \left[\alpha\left(E' - E_{0,\gamma}\right) + \beta\right]$$

where T_{c}, n, α and β are parameters that are specified by **TC**, **EXPONENT**, **SLOPE,** and **EDCONST,** respectively; E' is the initial internal energy; and $E_{0,\gamma}$ is the 0 K energy (electronic energy + zero-point energy) of γ-th isomer. We assume all isomers share the same **TC**, **EXPONENT**, **SLOPE** and **EDCONST**. This is compulsory in the **PARAMETER** block.

Available subsections and subkeywords: **SLOPE, EDCONST, TC, EXPONENT**

Example:

EDOWN
  SLOPE *0.00123*
  EDCONST[cm-1] *160*
  TC[K] *700*
  EXPONENT *1.0*
END_EDOWN

SLOPE

Description: Subkeyword for the parameter α. If users want to assume that $\langle \Delta E_{\mathrm{d}} \rangle$ is independent of internal energy, they can set $\alpha = 0$. Note that α is unitless. The default value is 0.

Available values: Float

Example: **SLOPE** *0.00123*

EDCONST

Description: Subkeyword for the parameter β. Note that β has units of energy.

This is compulsory in the **EDOWN** subblock.

Available values: Positive float

Example: **EDCONST**[cm-1] *130*

TC

Description: Subkeyword for the parameter T_c. The default value is 300 K

Available values: Positive float

Example: **TC**[K] *1600*

EXPONENT

Description: Subkeyword for the parameter n. If a user wants to use a temperature-independent $\langle \Delta E_d \rangle$, this can be done by setting $n = 0$. The default value is 0.

Available values: Float

Example: **EXPONENT** *1.0*

TEMPERATURE

Description: Subkeyword for temperatures. This is compulsory in **PARAMETER** block.

Available values: Positive float list

Example: **TEMPERATURE**[K]
100. 200. 300.
400. 500.
……
END_TEMPERATURE

PRESSURE

Description: Subkeyword for pressures. This is compulsory in **PARAMETER** block.

Available values: Positive float list

Example: **PRESSURE**[torr]
1E-6. 1E-5. 1E-4.
1E1. 1E3.
END_PRESSURE

GROUNDSPECIES

Description: Subkeyword for the name of an isomer or a bimolecular pair, of which the ground-state energy (energy at 0 K, by which we mean that vibrational zero-point energy is included) will be set as the zero of energy of all species. This is compulsory in the **PARAMETER** block.

Available values: String

Example: **GROUNDSPECIES** *Well_1*

5.3 Keywords in the REACTION block

Table 5. Glossary for the REACTION block

REACTION	
Description:	Start-keyword for a block describing all elementary reactions. This is compulsory in the standard input file.
Available sections	**BARRIERRXN, BARRIERLESSRXN**
Example:	**REACTION** **BARRIERRXN** **INFO** *Well_1-TS_1-Bim_1* **……** **END_BARRIERRXN** **……** **END_REACTION**
BARRIERRXN	
Description:	Start-keyword of a section for the elementary reaction with a barrier.
Available sections and subkeywords:	**INFO, PYRFILE, VARIATION, TUNNELING**
Example:	**BARRIERRXN** **INFO** *Well_1-TS_1-Bim_1* **END_ BARRIERRXN**
BARRIERLESSRXN	
Description:	Start-keyword of a section for a barrierless elementary reaction. Only works for bimolecular barrierless reactions.
Available subkeywords:	**INFO, RXNENERGY, AVGDIAMETER, RXNENERGY**
Example:	**BARRIERLESSRXN** **INFO** *Well_1-Bim_1* **AVGDIAMETER**[A] *4.0* **RXNENERGY**[kcal/mol] *-5.0* **END_ BARRIERLESSRXN**
INFO	
Description:	Subkeywords for the information of a reaction with a barrier or a barrierless elementary reaction. For an elementary reaction with a barrier, the value of this keyword should be the reactant name plus "-" plus the transition state name plus "-" plus the product name; for a barrierless elementary reaction, the value of this keyword should be the reactant name plus "-" plus the product name. If **PYRFILE** is also defined, the **INFO** will be used to name the species read from the *Polyrate* file. For a bimolecular pair, the first member is named with the name of bimolecular pair followed by a string "_1", and the second member is named with the name of bimolecular pair followed by a string "_2". Spaces

are not allowed in string values. This is compulsory in **BARRIERRXN** and **BARRIERLESSRXN**.

Available value: String

Example: **INFO** *Well_1-TS_12-Well_2* (for barrier elementary reaction)
INFO *Bim_1-Well_1* (for barrierless elementary reaction)

PYRFILE

Description: Subkeyword for the name of the *Polyrate* long output file for an elementary reaction that has an intrinsic barrier. If specified, the reactant, transition state, and product specified are read from the *Polyrate* output file and their names will be specified according to **INFO**. The user should make sure that the reactant, transition state, and product in **INFO** are consistent with those in **PYRFILE**. Section 4.5 gives a detailed description of the *Polyrate* long output file.

Available value: String

Example: **PYRFILE** *rxn_1.fu6*

AVGDIAMETER

Description: Subkeyword for the average diameter of the bimolecular reactant pair in a barrierless reaction. This is compulsory in the **BARRIERLESSRXN** section. If diameters of two species of the bimolecular are d_1 and d_2, then the average diameter is

$$d_{\mathrm{Avg}} = \frac{d_1 + d_2}{2}$$

Available value: Positive float

Example: **AVGDIAMETER**[A] *4.0*

RXNENERGY

Description: Subkeyword for the 0 K reaction energy of a barrierless reaction, which equals the ground-state energy of the product (including vibrational zero-point energy) minus the ground-state energy of the reactant (including vibrational zero-point energy) This is compulsory in the **BARRIERLESSRXN** section.

Available value: Float

Example: **RXNENERGY** [kcal/mol] *-5.0*

VARIATION

Description: Subkeyword for the method to consider the variational effect. We need the density of states (DoS) of a variational transition state along the reaction coordinate either to calculate the cumulative reaction probability (CRP) or the sum of states (SoS) during the calculation of the microcanonical flux coefficient. The **VARIATION** subkeyword will determine which DoS will be chosen.

If **VARIATION** is *tst*, the conventional transition state (which is a dividing surface passing through the saddle point) will be used.

If **VARIATION** is *cvt*, then – for each temperature – the program places the variational transition state at the point along the reaction

path that maximizes Gibbs free energy of activation.

If **VARIATION** is *muvt*, then – for each energy bin – the program places the variational transition state at the point along the reaction path that minimizes the CRP or sum of states.

The values *cvt* and *muvt* will work only if **PYRFILE** is specified. The default value is *tst*.

Available value: *tst, cvt, muvt*

Example: **VARIATION** *cvt*

TUNNELING

Description: Subkeyword for the method to consider the tunneling effect. The available values are *Eckart*, *zct* and *sct*.

If *zct* specified, the transmission probability will be loaded from the zero-curvature tunneling probability in the *Polyrate* output file.

If s*ct* specified, the transmission probability will be loaded from the small-curvature tunneling probability in the *Polyrate* output file.

If *Eckart* specified, the enthalpy of activation profile at 0 K will be fitted by an asymmetric Eckart potential; please refer to ref. 28 for details, but notice that ref. 28 did not include zero-point energies. Note that for gas-phase species at 0 K, the enthalpy equals the energy, and both equal the potential energy plus the zero-point energy.

The values *zct* and *sct* will work only if **PYRFILE** is specified. If the **TUNNELING** subkeyword is not specified, no tunneling effect will be included.

Available value: *Eckart*, *zct, sct*

Example: **TUNNELING** *sct*

INVLAPLACE

Description: Start-keyword for the subsection of inverse-Laplace-transform. We use a single-exponential or biexponential expression to fit the canonical flux coefficients(or so-called high-pressure-limit rate constant). Taking the biexponential expression as an example. We can express the canonical flux coefficients as

$$\hat{k} = A_1 \left(\frac{T}{T_1}\right)^{n_1} e^{-\frac{E_{a_1}}{k_B T}} + A_2 \left(\frac{T}{T_2}\right)^{n_2} e^{-\frac{E_{a_2}}{k_B T}}$$

The microcanonical flux coefficients can be obtained from the inverse-Laplace-transform to the canonical flux coefficient.

For reaction from bimolecular pair ν to isomer γ, there will be

$$\Delta\hat{k}(\nu \to \gamma | E) = \frac{e^{-\beta E}\Delta E}{\Phi_{\mathrm{rel}} Q_\nu^{\mathrm{SSHO}}} \left[\frac{\frac{1}{h^3}(2\pi m_\nu)^{\frac{3}{2}} A_1 \beta_1^{n_1}}{\Gamma\left(n_1 + \frac{3}{2}\right)} \int_0^E \rho_\nu(E_\eta - \varepsilon)(\varepsilon - E_{a_1})^{n_1 + \frac{1}{2}} \theta(\varepsilon - E_{a_1})\, \mathrm{d}\varepsilon + \frac{\frac{1}{h^3}(2\pi m_\nu)^{\frac{3}{2}} A_2 \beta_2^{n_2}}{\Gamma\left(n_2 + \frac{3}{2}\right)} \int_0^{E_\eta} \rho_\nu(E_\eta - \varepsilon)(\varepsilon - E_{a_2})^{n_2 + \frac{1}{2}} \theta(\varepsilon - E_{a_2})\, \mathrm{d}\varepsilon \right]$$

And the reverse reaction can be estimated from the detailed

balance.

For reaction from isomer γ to species ϕ (could be either unimolecular or bimolecular), there will be

$$\hat{k}(\gamma \to \phi|E) = \frac{A_1\beta_1^{n_1}}{\rho_\gamma(E)\Gamma(n_1)}\int_0^E \rho_\gamma(E-\varepsilon)\left(\varepsilon - E_{a_1}\right)^{n_1-1}\theta\left(\varepsilon - E_{a_1}\right)\mathrm{d}\varepsilon$$
$$+\frac{A_1\beta_1^{n_2}}{\rho_\gamma(E)\Gamma(n_2)}\int_0^E \rho_\gamma(E-\varepsilon)\left(\varepsilon - E_{a_2}\right)^{n_2-1}\theta(\varepsilon - E_{a_2})\,\mathrm{d}\varepsilon$$

And the reverse reaction can be estimated from the detailed balance. This is optional in the **BARRIERRXN** and **BARRIERLESSRXN** section.

Available value: **EXPRESSION, PREFACTOR1, T1, N1, EA1, PREFACTOR2, T2, N2, EA2**

Example:

INVLAPLACE
EXPRESSION *biexp*
PREFACTOR1[1/s] *10*
T1[K] *300*
N1 *1*
EA1[kcal/mol] *2*
PREFACTOR2[1/s] *100*
T2[K] *600*
N2 *1.2*
EA2[kcal/mol] *3.1*
END_INVLAPLACE

EXPRESSION

Description: Subkeyword for type of expression (single-exponential or biexponential) for fitting canonical flux coefficients. This is compulsory in the **INVLAPLACE** subsection.

Available value: *uniexp* or *biexp*

Example: **EXPRESSION** *biexp*

PREFACTOR1

Description: Subkeyword for A_1. The unit of it should be [1/s] for unimolecular reaction and [cm3/molecule/s] for bimolecular reaction. This is compulsory in the **INVLAPLACE** subsection.

Available value: Positive float

Example: **PREFACTOR1**[1/s] 10

T1

Description: Subkeyword for T_1. It has the unit of temperature. This is compulsory in the **INVLAPLACE** subsection both for **EXPRESSION** being *uniexp* or *biexp*.

Available value: Positive float

Example: **T1**[K] 300

N1

Description: Subkeyword for n_1. It is unitless. This is compulsory in the **INVLAPLACE** subsection both for **EXPRESSION** being *uniexp*

	or *biexp*.
Available value:	Positive float
Example:	**N1** 1.0

EA1

Description:	Subkeyword for E_{a_1}. It has the unit of energy. This is compulsory in the **INVLAPLACE** subsection both for **EXPRESSION** being *uniexp* or *biexp*.
Available value:	Float
Example:	**EA1**[kcal/mol] 2.3

PREFACTOR2

Description:	Subkeyword for A_2. The unit of it should be [1/s] for unimolecular reaction and [cm3/molecule/s] for bimolecular reaction. This is compulsory in the **INVLAPLACE** subsection for **EXPRESSION** being *biexp*.
Available value:	Positive float
Example:	**PREFACTOR2**[1/s] 10

T2

Description:	Subkeyword for T_2. It has the unit of temperature. This is compulsory in the **INVLAPLACE** subsection for **EXPRESSION** being *biexp*.
Available value:	Positive float
Example:	**T2**[K] 300

N2

Description:	Subkeyword for n_2. It is unitless. This is compulsory in the **INVLAPLACE** subsection for **EXPRESSION** being *biexp*.
Available value:	Positive float
Example:	**N2** 1.0

EA2

Description:	Subkeyword for E_{a_2}. It has the unit of energy. This is compulsory in the **INVLAPLACE** subsection for **EXPRESSION** being *biexp*.
Available value:	Float
Example:	**EA2**[kcal/mol] 2.3

5.4 Keywords in the SPECIES block

Table 6. Glossary for in the SPECIES block

SPECIES	
Description:	Start-keyword of species block. This is not compulsory in the standard input file. If all species in the master equation are read from *Polyrate* long output files, the **SPECIES** block is not necessary. When the *Polyrate* long output file is defined, the **SPECIES** block can still be defined to give some supplementary information about species. The **SPECIES** block has three kinds of sections: one kind for isomers, which are the wells and are specified in **WELL** sections, one kind for transition states, which are specified in **TRANSTATE** sections, and a third kind for bimolecular pairs, which are specified in **BIM** sections.
Available sections:	**WELL, TRANSTATE, BIM**
Example:	**SPECIES** **WELL** **……** **END_WELL** **……** **END_SPECIES**
WELL	
Description:	Start-keyword of well section. In the input for a reaction system, there must be at least one well (in this manual, well and isomer are synonyms); thus, if no well is defined in *Polyrate* output files, **WELL** is compulsory in the **SPECIES** block and can appear multiple times.
Available Subkeywords:	**NAME, GEOMETRY, FREQUENCY, ELELEVEL, E0K, EELE, ROTSIGMA, OPTICALNUM, BMAX, DIAMETER, EPSILON, G09FILE, MSTFILE**
Example:	**WELL** **GEOMETRY**[A] *C 1.223 2.334 3.4456* …… **END_GEOMETRY** **……** **END_WELL**
TRANSTATE	
Description:	Start-keyword of transition state section. **TRANSTATE** is not compulsory in the **SPECIES** block.
Available Subkeywords:	**NAME, GEOMETRY, FREQUENCY, ELELEVEL, E0K, EELE, ROTSIGMA, OPTICALNUM, IMAGFREQ, G09FILE, MSTFILE**

Example:

```
TRANSTATE
    GEOMETRY[A]
        C 1.223 2.334 3.4456
        ……
    END_GEOMETRY
    ……
END_ TRANSTATE
```

BIM

Description: Start-keyword of bimolecular pairs section. A bimolecular pair contains two species, which can be either atoms or molecules. The first species should be defined by **SP1MOL** or **SP1ATOM**; the second species should be defined by **SP2MOL** or **SP2ATOM**. It does not matter which one appears first.

Available subsections and subkeywords: **SP1MOL**, **SP2MOL**, **SP1ATOM**, **SP2ATOM**, **EELE**, **E0K**, **NAME**

Example:

```
BIM
    SP1MOL
      ……
    END_SP1MOL
    SP2ATOM
      ……
    END_SP2ATOM
END_ BIM
```

SP1MOL, **SP2MOL**

Description: Start-keyword of one of the molecules (the number of atoms should greater than 1) in bimolecular pair section. “SP1” denotes species 1 and choosing which one as species 1 does not matter.

Available subkeywords: **NAME, GEOMETRY, FREQUENCY, ELELEVEL, E0K, EELE, ROTSIGMA, OPTICALNUM, G09FILE, MSTFILE**

Example:

```
SP1MOL
   ……
END_SP1MOL
```

SP1ATOM, **SP2ATOM**

Description: Start-keyword of one of the atoms in a bimolecular pair section. “SP1” denotes species 1, and it does not matter which one is chosen as species 1.

Available subkeywords: **NAME, SYMBOL, ELELEVEL, E0K, EELi**

Example:

```
SP1ATOM
   ……
END_SP1ATOM
```

NAME

Description: Subkeyword for the name of species. This is compulsory in every **WELL**, **TRANSTATE**, or **BIM** section and also in the **SP1MOL**, **SP2MOL**, **SP1ATOM**, and **SP2TAOM** subsections. Spaces are not allowed in name strings. The length of a name has no limitation, but for output aesthetics we recommend it not contain more than 15 characters.

Available value: String

Example 1: **NAME** Well_1

Example 2: **NAME** 1-propyl

GEOMETRY

Description: Subkeyword for the geometry of a species. This is not compulsory when the *Polyrate* output file or *Gaussian* output file is defined. There is no requirement on conditions that must be satisfied by the geometry; for example, it is not necessary to put the center of mass at the origin. Furthermore, there is no requirement on how the geometries of different species are related, even for coordinates of the two members of a bimolecular pair. The coordinates given here are not manipulated in the program.

Available value: List. Each line should contain four entries. The first is the atom symbol string; the other three are float numbers.

Example:

GEOMETRY[A]
C 1.234 2.345 3.456
C 4.567 5.678 6.789
H 7.890 8.987 1.670
END_GEOMETRY

FREQUENCY

Description: Subkeyword for the real frequencies of a species. For a linear molecule, the user should input $3N-5$ frequencies; for a nonlinear molecule, user should input $3N-6$ frequencies. For a linear transition state, user should input $3N-6$ frequencies; for a nonlinear transition state, user should input $3N-7$ frequencies. The program will automatically determine whether a molecule or a transition state is linear. This keyword is not compulsory when the *Polyrate* output file or *Gaussian* output file is defined.

Available value: Positive float list

Example:

FREQUENCY[cm-1]
3585.12 3506.34 1885.23
45.49 235.67 478.87
END_FREQUENCY

ELELEVEL

Description: Subkeyword for the electronic energy level of species. The input can contain multiple lines, and each line should contain two number. The first number denotes degeneracy, and the second

number denotes the energy relative to the electronical ground energy. The default value is *1 0.*

Available value: Positive float list

Example:
ELELEVEL[a.u.]
2 0
2 0.023
END_ELELEVEL

E0K

Description: Subkeyword for the energy of species at 0 K, which equals the electronic energy (including, as usual, the nuclear repulsion) plus the zero-point vibrational energy. Note that for gas-phase species, the 0 K energy is the same as the 0 K enthalpy.

This subkeyword is not compulsory when the *Polyrate* output file or *Gaussian* output file is defined for this species. If some species energies are defined by **E0K** subkeywords and some are read by *Gaussian*, please make sure that their zeros of energy are consistent.

Available value: Float

Example: **E0K**[kcal/mol] *10.0*

EELE

Description: Subkeyword for the classical energy of the species at 0 K, which equals the electronic energy (the electronic energy always includes the nuclear repulsion). This is not compulsory when a *Polyrate* output file or *Gaussian* output file is defined for this species. If some species energies are defined by **EELE** subkeywords and some are specified by *Gaussian*, the user should make sure that their zeros of energy are consistent.

Available value: Float

Example: **EELE**[kcal/mol] *10.0*

ROTSIGMA

Description: Subkeyword for the rotational symmetry number of a species. The electronic-vibrational-rotational partition function, the sum of states, and the density of states will be divided by this number. The default value is 1. If the properties of a species are defined by a Gaussian output file, the program will automatically read **ROTSIGMA** from that file, and that will override the default. However, if the properties of a species are read from a Polyrate output file, the program will not be able to read **ROTSIGMA** because Polyrate output files do not include **ROTSIGMA**. Therefore, if one is reading the properties of a species from a Polyrate output file, and if the user does not want the rotational symmetry number of the species to be unity, one must use the **ROTSIGMA** keyword to set it to another value.

Available value: Positive integer

Example: **ROTSIGMA** *2*

OPTICALNUM

Description: Subkeyword for the optical-isomer number α of a species; this is 2 for a chiral species and 1 for a species that can be rotated to coincide with its mirror image. The value of α will only be used when calculating the density of states, but it will be automatically included in variables derived from the density of states, e.g., the microcanonical flux coefficient and the sum of states. The default value is 1.

Polyrate and *Gaussian* outputs do not include the optical-isomer number, so you must define it if you want to assign a non-unit optical isomer number to a species.

Note that the MS-T method properly accounts for optical isomers,[29] so if one uses MS-T, one should not use α.

Available value: Positive integer

Example: **OPTICALNUM** *2*

DIAMETER

Description: Subkeyword for the diameter of an isomer. This is compulsory for each **WELL** when **LJCOLLISON** is defined and the **DIAMETERA** is undefined in **LJCOLLISION**. When **DIAMETER** is defined in **WELL** and **DIAMETERA** is also defined in **LJCOLLISION**, **DIAMETER** will cover **DIAMETERA.**

Available value: Positive float

Example: **DIAMETER**[A] *4.0*

EPSILON

Description: Subkeyword for the Lennard-Jones energy parameter of an isomer. This is compulsory for each **WELL** when **LJCOLLISON** is defined and **EPSILONA** is undefined. When both **EPSILON** is defined in **WELL**, and **EPSILONA** is also defined in **LJCOLLISION**, **EPSILON** will cover **EPSILONA.**

Available value: Positive float

Example: **EPSILON** [cm-1] *40.*

FREQSCALE

Description: Subkeyword for the scaling factor of frequencies. The code has three possible ways to get frequencies for a species: from the *Polyrate* standard output, from the *Gaussian* standard output, or from the **FREQUENCY** subkeyword. If scaled frequencies are used in the *Polyrate* run, the frequencies extracted from the *Polyrate* output are already scaled, but if *Gaussian* was not asked to scale the frequencies, then *Gaussian* will have outputted unscaled frequencies. This keyword only works for frequencies read from *Gaussian* standard output and from the **FREQUENCY** subkeyword. Our recommendation to the user is that frequencies should always be scaled to correct for anharmonicity and for

systematic errors in electronic structure calculations;[30] thus one of these options should be used (and the user should be sure that the frequencies are scaled only once). The default value of this keyword is 1.0.

Available value: Positive float

Example: **FREQSCALE** *0.972*

G09FILE

Description: Subkeyword for the name of the *Gaussian 09* output file for a species. Section 4.5 gives a detailed description of the *Gaussian* long output file. If specified, then geometries, frequencies, symbols, rotational symmetry number, imaginary frequency and energies will be read from the *Gaussian* output file.

Available value: String

Example: **G09FILE** *Well_1.log*

MSTFILE

Description: Subkeyword for the name of the *MSTor* output file for a species. If specified, the MST density of state will be read from this file. This is optional. Section 4.5 will give a detailed description of the *MSTor* long output file.

Available value: String

Example: **MSTFILE** *Well_1.out*

5.5 *Polyrate*, *Gaussian* and *MSTor* Output Files

5.5.1 *Polyrate* output file

The *Polyrate* output file specified in keyword **PYRFILE** is the standard output file *xxx.fu6* of *Polyrate*. The species information, the minimum energy path (MEP) information, and the transmission possibilities are read from this file. The *Polyrate* version should be 2016 or later. The keyword "PRINTSTEP" should be specified in *Polyrate* when it is used in conjunction with TUMME. In case of compatibility issues, the user may want to check the output file of *Polyrate*; below we list the key-strings in the *Polyrate* output file that TUMME uses to look for the values. Note that the *Polyrate* program can be used with many interfaces – see http://truhlar.chem.umn.edu/content/software – but the choice of interface does not affect the essential elements of the *Polyrate* long output, so TUMME should work for every available electronic structure interface of *Polyrate*, including the very popular *Gaussrate*. When the code is reading the standard output file of *Polyrate*, the code will try to read all the values listed in Table 7. TUMME does not let user choose which values to read. The code will scan the file from the first line to the last line; if any of key-strings showed in Table 7 is met, the corresponding value will be read. So, the user should make sure that the *Polyrate* standard output file being used contains the generalized transition states if the user sets the **VARIATION** keyword as *cvt* or *muvt*; and the user should make sure that the transmission probability is in the *Polyrate* output file if the user sets **TUNNELING** as *sct* or *zct*.

If *Polyrate* uses scaled frequencies, the frequencies at stationary points or generalized transition states in the *Polyrate* standard output are always the scaled ones, so we do not scale frequencies again in TUMME.

Table 7. Keywords in *Polyrate* output file

Values	Key-strings	Description
Atom symbols	"Atomic information:"	Atomic symbols will appear from the third line after the key-string line. The first entry of each line is the sequence number of each atom. The third entry of each line is the atom symbol. They will both be loaded. The reading will end with a blank line.
Temporary species objects	"Reactant #1"// "Reactant #2"// "Product #1"// "Product #1"	When one of four keywords is recognized in the output file, a temporary species object will be created. Then the species information will temporarily be stored in these objects and after all information read, the reactant, transition state and product objects will finally be created.
Geometries	"********** Reactants:"// "********** Products:"// "********* Saddle point:"	Geometries will appear from the fifth line after the key-string line and the last three entries of each line are coordinates in angstrom.
Species properties	"****** Reactant 1 **********" // "****** Reactant 2 **********" // "****** Product 1 **********" // "****** Product 2 **********" // "****** Saddle point **********"	Number of atoms will appear in the second line after the key-string line, shown as "xx-atom" where "xx" denotes an integer; then electronic degeneracies and energies appear in the same line beginning with string "electronic degeneracies and energies (a.u.)" ; then frequencies will appear beginning from the fourth line after the line containing string "Harmonic Frequencies"

Reaction energy	"Reaction energetics"	The reaction energy calibrated with ZPVE in atomic unit is the seventh entry of the 16-th line after the key-string line. This is the reaction enthalpy at 0 K.
Barrier energy	"V+ZPE w/re reactant V+ZPE"	The barrier energy including the change of ZPVE will appear in the same line. This is the enthalpy of activation at 0 K.
Geometries of generalized transition states in MEP	"Space-fixed cartesian coordinates vs reaction coordinate"	The first geometry of the generalized transition state will appear from the fifth line after the key-string line and the last three entries of each line are coordinates; after *Natom* lines, skipping three lines, the geometries of the next generalized transition state will begin. This will circulate again and again until the string "*Classical and adiabatic energies*" is met. The reaction coordinate *s* will also be recorded.
Frequencies and energies of generalized transition states in MEP	"s(angstrom) VMEP Va^G mu^CD-SC frequencies"	After a blank line, the frequencies and energies will appear. The sum of the potential energy and the ground-state local vibrational energy (ZPVE) in modes normal to the reaction path is called V_a^G (which denotes vibrationally adiabatic ground-state potential energy curve;[18] there are $3N-7$ (for a nonlinear species) or $3N-6$ (for a linear species) frequencies and they will be read until a blank line is encountered. The V_a^G will also be read for each value of the reaction coordinate *s*.
Transmission probability	"Transmission probabilities"	The transmission probability will appear beginning at the fourth lines after the key-string line. If the elementary reaction is barrierless, a string "The Classical barrier is less than zero!" should be found. The code will try to find the SCT probability if the user has chosen *sct* and will try to find the ZCT probability if the user has chosen *zct*. If found, the corresponding transmission probability will be read, otherwise this will default to a None-type in Python.

5.5.2 *Gaussian* output file

The *Gaussian* output file specified in keyword **G09FILE** is the standard *Gaussian* output file. When specified for a species, the geometries, frequencies, rotational symmetry number, and energy at 0 K will be read from the file. Note that "energy at 0 K" refers to the energy including zero-point energy; this can also be called the enthalpy at 0 K. The *Gaussian* version should be *Gaussian 09 or Gaussian 16*. The keyword "freq" should be specified in the *Gaussian* input file. Here we list the key-strings in the *Gaussian* output file used to look for the values.

If frequencies read from the *Gaussian* standard output file are unscaled, the user should set the **FREQSCALE** subkeyword.

Table 8. Keywords in *Gaussian* output file

Values	Key-strings	Description
Geometry	The last string "`Standard orientation`"	Geometries will appear in 4 lines afterwards. The second entry of each line are atom number and the last three are coordinate in angstroms. Read will end with a line full of "----".
Frequency	"`Frequencies --`"	Frequencies will appear on the same line that starts with the key-string.
Rotational symmetry number	"`Rotational symmetry number`"	Rotational symmetry number will appear in the same line afterwards
E0K	"`Sum of electronic and zero-point Energies`"	E0K will appear in the same line afterwards

5.5.3 *MSTor* output file

The *MSTor* output file specified in keyword **MSTFILE** is the standard *MSTor* output file. When specified for a species, the density of states will be read from the file. The *MSTor* version should be 2017 or later.

The keywords "**estep**" and "**emax**" should be specified in the *MSTor* input file. These two keywords are not illustrated in the manual of *MSTor*2017: here we give an explanation of these two keywords.

estep is the energy step used in calculating the density of states. The unit is kcal/mol. It should be specified in the $GENERAL section of a *MSTor* input file. Notice that the energy step **estep** in *MSTor* should be the same as the energy step **DE** in TUMME.

emax is the maximum energy of energy level used for calculating the density of states. The unit is kcal/mol. Should be specified in $GENERAL section of a *MSTor* input file. Notice that the maximum energy **emax** in *MSTor* should be the same as the energy step **EMAX** in TUMME.

6. Detailed Implementation

6.1 Overview of all source files

Table 9 Description of Python source files

File Name	Description
parallel_lib.py	This file contains a *Mpv_class* class which stores the variables like process rank number and total processor number for MP parallel mode. This file also contains a method to gather the stack of *ME_class* of each processor into rank 0 processor. We do not simply use the built-in *gather()* function in *mpi4py* because in this function there's memory limitation (< 2GB) for the communicated object.
data_lib.py	This file contains some functions dealing with the "not a number" issue of the density of states in *MSTor* output file and the program will use a spline function to fit the density of states.
dd_lib.py	This file contains an interface function for Python to call the quadruple-precision C++ dynamical library.
qd_lib.py	This file contains an interface function for Python to call the octuple-precision C++ dynamical library.
const_lib.py	This file contains physical constants, unit conversion constants, and the like.
global_lib.py	This file contains global variables, keywords for standard input file and default values for global variables.
molecule_lib.py	This file contains *Molecule_class* class and *Atom_class* class. They are father classes for species classes. The density of states and partition function can be calculated in the two classes. They contain some basic properties of a molecule or transition state.
species_lib.py	This file contains classes for wells, transition states and fragments of a bimolecular pair. (In this manual, well and isomer are synonyms.) The relaxation kernel and Boltzmann population can be calculated in the class for wells. They are children classes of the father classes in *molecule_lib.py*.
reaction_lib.py	This file contains *BarrierRxn_class* and *BarrierlessRxn_class* two classes. They both have method to estimate the microcanonical flux coefficient. *BarrierRxn_class* contains an object of *Mep_class* which will load the minimum energy path, stationary species, transmission probability from a *Polyrate* output file.
mep_lib.py	This file contains a *Mep_class* class to extract the minimum energy path (MEP), stationary species, transmission probability from a *Polyrate* file.

collision_lib.py	This file contains *HS_collision_class* and *LJ_collision_class*. The *HS_collision_class* is to store the collision information of the hard-sphere collision model. The *LJ_collision_class* is to store the collision information of the Lennard-Jones collision.
me_lib.py	This file contains a *ME_class* class to store all the information about a master equation for a given temperature and pressure. It has a method to construct a transition matrix, check whether a merge has occurred, and derive the phenomenological rate constants. For every working condition (T, p), a *ME_class* class will be created. After solving for the phenomenological rate constants, the memory of most of the variables will not be released. The *ME_class* object will be pushed into a stack. For running in parallel, after all calculations are finished, the stack will be gathered into rank 0. Then rank 0 will deal with the printout of the rate constant/ eigenpair/microcanonical flux coefficient/partition function/time evolution.
output_lib.py	This file contains a *Std_out_class*, *Serial_out_class*, *Mp_out_class*, and *Mpi_out_class* classes, corresponding to the serial, MP, and MPI schemes. Due to the disparity of the printout arrangements between serial and parallel schemes, to make the printout compatible in serial and parallel schemes, we set a string stack. All the output information during a run is pushed into the string stack. When all the calculations are finished, the string stack is gathered into rank 0 and printed out to output files.
tumme_main.py	This file contains *serial_main*, *mp_main*, and *mpi_main,* which are three kinds of main routines for different run modes.
tumme_readin.py	This file contains two functions named *readParallel* (to read two parallelism subkeywords) and *std_readin* (to read other information including global parameters, elementary reactions, and species properties).
tumme_pre.py	This file contains *std_pre*, *serial_pre*, *mp_pre,* and *mpi_pre* functions for initialization of different types of runs. *std_pre* is the common code shared by serial, MP, and MPI runs. In the *std_pre* function, the energies of all species will be placed relative the 0 K energy set by **GROUNDSPECIES**; the density of states of all species will be set; and a reaction map will be created. A reaction map is a matrix in which the (i, j) element is an object of the reaction classes in *reaction_lib.py* if the i-th species and j-th species are connected by an elementary reaction; otherwise (i, j) element is a None type of Python.

tumme_solver.py	This file contains *std_solver*, *serial_solver*, *mp_solver,* and *mpi_solver* functions for different solvers for different runs. *std_solver* is the common code shared by serial, MP and MPI runs. In the *mp_solver* and *mpi_solver* function, the entire set of temperature and pressure conditions {(*T*,*p*)} will be divided into *nproc* subsets. Each rank of the processor will run its own subset. Finally, all the information will be gathered into the rank 0 processor and printed out there. ,

Table 10 Description of C++ Sources files

File Name	Description
Interface_dd.h *Interface_dd.cpp* *ME_solver_dd.h* *ME_solver_dd.cpp*	These files contain quadruple precision code of the standard process in Fig. 1. The relaxation kernel for all isomers and the symmetric transition matrix will be constructed and diagonalized in quadruple precision; then the quadruple-precision eigenpairs will be transformed into double precision and returned to Python. OpenMP is utilized in the C++ code to reduce the computation cost.
Interface_qd.h *Interface_qd.cpp* *ME_solver_qd.h* *ME_solver_qd.cpp*	These files contain octuple-precision of code of the standard process in Fig. 1. The relaxation kernel for all isomers and the symmetric transition matrix will be constructed and diagonalized in octuple precision; then the octuple-precision eigenpairs will be transformed into double precision and returned to Python. OpenMP is utilized in the C++ code to reduce computation cost.
qd/	Original source code for *qd* library.
mpack/	Simplified source code for *mpack* library.

6.2 Normalization of the energy transfer kernel

The energy transfers are assumed to be governed by the exponential down model, which assumes that the probability that the isomer energy changes from E' to E in a single collision with the bath gas is

$$P_\gamma(E|E') = A(E')e^{-\theta(E')\cdot(E'-E)} \qquad E' \geq E \tag{31}$$

where A is a normalization constant, and θ is a collision efficiency parameter. The probability for $E' < E$ can be determined by the detailed balance, yielding

$$P_\gamma(E|E') = \begin{cases} A(E')e^{-\theta(E')\cdot(E'-E)} & E' \geq E \\ A(E)\dfrac{f_\gamma(E)}{f_\gamma(E')}e^{-\theta(E)\cdot(E-E')} & E' < E \end{cases} \tag{2}$$

where $f_\gamma(E)$ is the thermal population density $\rho_\gamma(E)e^{-\beta E}$ of isomer γ with energy E.

After discretization, Eq. (2) becomes

$$P_\gamma[\eta,\eta'] = \begin{cases} A[\eta']e^{-\theta(E_{\eta'})\cdot(E_{\eta'}-E_\eta)} & E_{\eta'} \geq E_\eta \\ A[\eta]\dfrac{f_\gamma[\eta]}{f_\gamma[\eta']}e^{-\theta(E_\eta)\cdot(E_\eta-E_{\eta'})} & E_{\eta'} < E_\eta \end{cases} \tag{3}$$

where η and η' are indices of energy bins, and the bins are arranged in order of decreasing energy so $\eta = 1$ labels the highest-energy bin, and $\eta = N_\gamma$ denotes the zero-point energy of isomer species γ, where N_γ is the number of energy bins used for the γ-th isomer.

The normalization condition is

$$\Sigma_\eta P_\gamma[\eta,\eta'] = 1 \tag{4}$$

For $\eta' = 1$, $\Sigma_\eta P[\eta,1] = 1$, and Eq. (4) gives

$$\sum_{\eta=1}^{N_\gamma} A[1]e^{-\theta(E_1)\cdot(E_1-E_\eta)} = 1 \tag{5}$$

and

$$A[1] = \frac{1}{\sum_{\eta=1}^{N_\gamma} e^{-\theta(E_1)\cdot(E_1-E_\eta)}} \tag{6}$$

For $\eta' = 2$,

$$A[1]\frac{f_\gamma[1]}{f_\gamma[2]}e^{-\theta(E_1)\cdot(E_1-E_2)} + \sum_{\eta=2}^{N_\gamma} A[2]e^{-\theta(E_2)\cdot(E_2-E_\eta)} = 1 \tag{7}$$

and therefore

$$A[2] = \frac{1 - A[1]\dfrac{f_\gamma[1]}{f_\gamma[2]}e^{-\theta(E_1)\cdot(E_1-E_2)}}{\sum_{\eta=2}^{N_\gamma} e^{-\theta(E_2)\cdot(E_2-E_\eta)}} \tag{8}$$

For general $\eta' = n$,

$$\sum_{\eta=1}^{n-1} A[\eta]\frac{f_\gamma[\eta]}{f_\gamma[n]}e^{-\theta(E_\eta)\cdot(E_\eta-E_n)} + \sum_{\eta=n}^{\eta=N_\gamma} A[n]e^{-\theta(E_n)\cdot(E_n-E_\eta)} = 1 \tag{9}$$

and therefore

$$A[n] = \frac{1 - \sum_{\eta=1}^{n-1} A[\eta]\dfrac{f_\gamma[\eta]}{f_\gamma[n]}e^{-\theta(E_\eta)\cdot(E_\eta-E_n)}}{\sum_{\eta=n}^{N_\gamma} e^{-\theta(E_n)\cdot(E_n-E_\eta)}} \tag{10}$$

In many cases, when n becomes large, the numerator of Eq. (10) becomes negative, for example at $n = N_t$. Then the energy levels $N_t + 1$, $N_t + 2$,…, N_γ are removed, and N_γ is decreased accordingly. After this removal, the normalization is re-calculated with the new, smaller N_γ. This process is repeated until the numerator of Eq. (10) is positive for all retained bins. This calculation is carried out in the file *species_lib.py*.

One finds that the average energy transferred per collision in the subset of collisions in which the isomer loses energy to the bath gas is approximately $1/\theta$ because

$$\frac{\int_{-\infty}^{E'}(E'-E)e^{-\theta(E'-E)}dE}{\int_{-\infty}^{E'}e^{-\theta(E'-E)}dE} = \frac{1}{\theta(E')} \tag{11}$$

Therefore, the program sets $\theta(E') = \frac{1}{\langle \Delta E_\mathrm{d} \rangle (E')}$ where $\langle \Delta E_\mathrm{d} \rangle (E')$ is set by the subsection **EDOWN** in the **PARAMETER** block.

6.3 Bound isomers to CSE modes one-to-one.

After constructing the symmetric transition matrix **G**, the program diagonalizes it to get eigenpairs. The eigenpairs are ordered in ascending order of eigenvalues. The λ-th eigenvector is denoted $\boldsymbol{u}^{(\lambda)}$. If a reaction system has S isomers, then there should be S CSE eigenvectors. We define the projection of $\boldsymbol{u}^{(\lambda)}$ on the chemical basis vector of isomer γ as EPCS:

$$EPCS_\gamma^{(\lambda)} = \frac{1}{\sqrt{Q_\gamma/\Delta E}} \sum_{i'=1}^{N_\mathrm{y}} \delta_{\gamma\gamma'} u_{i'}^{(\lambda)} F_{i'} \tag{12}$$

where $\delta_{\gamma\gamma'}$ is a Kronecker delta; $F_{i'}$ is $\sqrt{\rho_{\gamma'}(E_{\eta'})e^{-\beta E_{\eta'}}}$, which is the square root of the Boltzmann population of unimolecular species γ' with internal energy $E_{\eta'}$; Q_γ is the electronic-vibrational-rotational partition function of isomer γ; $u_{i'}^{(\lambda)}$ is the i-th element of the λ-th eigenvector; N_y is the size of the transition matrix; ΔE is the energy step between energy bins. Then we define the eigenvector projection squared on the relaxation subspace as EPSRS

$$EPSRS^{(\lambda)} = P^{(\lambda)} = 1 - \Sigma_\gamma \left[EPCS_\gamma^{(\lambda)} \right]^2 \tag{13}$$

The program will associate each of the CSE eigenmodes in a one-to-one relationship with a specific isomer. The algorithm to determine the correspondence is as follows.

For an eigenmode λ', the program looks for the maximum $\left| EPCS_\gamma^{(\lambda')} \right|$ among $\gamma = 1, \ldots, S$. Let γ' label the eigenmode thus located. This means the basis vector of isomer γ' has the greatest projection onto the eigenvector λ'. A correspondence tuple (λ', γ') that denotes the correspondence between isomer γ' and eigenmode λ' is assigned on this basis. When each isomer is associated with one eigenmode, one has achieved the desired one-to-one correspondence. The situation is more complicated in a case where an isomer corresponds to two or more eigenmodes. In order to make a one-to-one assignment for the eigenmode λ', the program may need to assign isomers iteratively. If an isomer has multiple associated eigenvectors, the program finds the eigenvector that has the maximum projection and creates a corresponding tuple. This may then force the reassignment of other eigenvectors by

selecting the second largest $\left|EPCS_{\gamma}^{(\lambda')}\right|$ (or third or fourth largest – this will continue until all eigenvectors have been associated with an isomer).

γ_3	**0.2**	**0.8**	**0.6**
γ_2	**0.2**	**0.1**	**0.5**
γ_1	**0.9**	**0.2**	**0.1**
	λ_1	λ_2	λ_3

Figure 3. The elements of the $\left|EPCS_{\gamma}^{(\lambda)}\right|$ matrix for an artificial reaction system with 3 isomers. The eigenvectors are numbered in ascending order of the eigenvalues; therefore λ_1, λ_2, and λ_3 are CSE modes. In CSE theory the sum of the squares of the elements of a column will be close to one, which means that EPSRS will be small. In this case, it is easily calculated from the value in the figure that EPSRS is 0.11, 0.31 and 0.38 respectively for the λ_1, λ_2 and λ_3 eigenmodes.

Let's take an artificial reaction system having three isomers as an example to illustrate the algorithm, and for this example we assume there is no merger. (The case of a merger is discussed in the next paragraph.) The relevant elements of $\left|EPCS_{\gamma}^{(\lambda)}\right|$ for this example are in Fig. 3. (Because this is a model system, the eigenvector components are assumed to be precisely the values stated, even though they have only one significant figure.) The eigenvectors are numbered in order of increasing eigenvalues, i.e., the slowest modes are assumed to be the CSE modes. The program always recognizes the first *S* eigenvectors as the CSEs, where *S* is the number of isomers. Thus, the three eigenmodes shown in Fig. 3 are the CSE modes. For eigenmode with eigenvalue λ_1, we find that the maximum $\left|EPCS_{\gamma}^{(\lambda)}\right|$ over $\gamma_1, \gamma_2,$ and γ_3 is for γ_1. Therefore, in this case the program would assign the first tuple as (λ_1, γ_1). For eigenmode 2 (i.e., the one with eigenvalue λ_2), it is then straightforward to assign the tuple (λ_2, γ_3), and for eigenmode λ_3, the tuple (λ_3, γ_3) is defined. At this stage, the isomer γ_3 corresponds two eigenmodes, λ_2 and λ_3. Since $\left|EPCS_{\gamma_3}^{(\lambda_2)}\right| > \left|EPCS_{\gamma_3}^{(\lambda_3)}\right|$, the isomer γ_3 is viewed as being associated with eigenmode λ_2, and the tuple (λ_3, γ_3) is accordingly deleted. Then the eigenmode λ_3 is searched for the second-highest value over γ_1, γ_2, and γ_3, which is $\left|EPCS_{\gamma_2}^{(\lambda_3)}\right|$, and therefore the tuple (λ_3, γ_2) is defined. The resultant set of tuples is (λ_1, γ_1), (λ_2, γ_3), and (λ_3, γ_2). Now every eigenmode has been assigned to an isomer, and the set of correspondence tuples satisfies the one-to-one condition. The ranking of eigenvectors in an ascending order of eigenvalues is carried out for each (*T*, *p*) combination, and the first *S* eigenvalues are recognized as CSE modes for that (*T*, *p*); then the one-to-one assignment algorithm shown above is executed for that (*T*, *p*) combination.

For cases where a merger occurs (see Section 5.4 for a discussion of mergers), there is a probability that eigenvalues whose eigenvectors correspond to chemical reaction processes are faster than some internal relaxation processes. If this were to happen, the assumption that the first *S* eigenmodes are CSE modes and others are IERE modes would be invalid. So far, though, no systematic way has been presented to handle this case, and therefore TUMME will always recognize the first *S* modes (in order of increasing eigenvalues) as the CSE ones. After ranking the eigenmodes, recognizing CSE modes, and binding isomers to CSE modes one-to-one, the program checks all CSE modes as to whether they are merged with IERE modes. If a CSE eigenvector does merge with the IERE space, this eigenmode will be moved from the CSE space into the IERE space, but the one-to-one assignment of other CSE modes will not be changed.

This part is done in the *me_lib.py* file.

6.4 Handling the merge condition

When temperatures are very high or pressures are very low, some CSE eigenmodes may merge with IERE eigenmodes. In this part of the manual, we give the details of how the program detects the merge condition and how it deals with it.

The value of $EPSRS^{(\lambda)}$ provides the criterion to judge whether a CSE eigenmode is merging with the IERE space. If this value is greater than **MERGETHRESHOLD**, the merge is assumed to have occurred; otherwise not.

If a CSE eigenmode is determined to have merged with the IERE space, the corresponding isomer is viewed as merging with some other species. There is no unambiguous prescription for assigning which species it has merged with. In TUMME, we adopted the viewpoint of Georgievskii et al. in their statement: "*For an eigenvector describing equilibration with bimolecular products, the macroscopic population for each unimolecular species has the same sign. For an eigenvector that describes equilibration between two groups of unimolecular species, the cumulative populations for those groups are generally approximately equal in magnitude and opposite in sign, at least at not too high a temperature.*"[5] For a merged eigenmode λ', let the one-to-one isomer determined by Section 5.3 be γ'; then the algorithm to determine the other merged species is as follows:

1) For all $\gamma = 1, \dots, S$, all positive $EPCS_{\gamma}^{(\lambda')}$ values are summed together, and all negative values are summed together. The absolute values of the two sums are compared. The larger one will be a normalization constant to scale the $EPCS_{\gamma}^{(\lambda')}$ for all $\gamma = 1, \dots, S$.
2) The program then finds the two largest absolute values of the normalized $EPCS_{\gamma}^{(\lambda')}$ over $\gamma = 1, \dots, S$; let the isomer index for them be γ_1 and γ_2. If the absolute value of the normalized $EPCS_{\gamma_1}^{(\lambda')}$ and $EPCS_{\gamma_2}^{(\lambda')}$ are both greater than 0.5 and $EPCS_{\gamma_1}^{(\lambda')}$ and $EPCS_{\gamma_2}^{(\lambda')}$ have different sign, this eigenmode will be recognized as the equilibration between isomer γ_1 and isomer γ_2. Further, if both γ_1 and γ_2 are not equal to γ', the programs reports an error and exits.

3) If one of the absolute values of the normalized $EPCS_{\gamma_1}^{(\lambda')}$ or $EPCS_{\gamma_2}^{(\lambda')}$ is less than 0.5, this eigenmode will be recognized as the equilibrium between isomer γ' and a bimolecular pair. The kappa matrix (defined next) will be analyzed to further determine which bimolecular pair it is.
4) The (γ, ν) element of the kappa matrix is defined as [5,25]

$$\kappa_{\gamma,\nu} = \sum_{\lambda=S+1}^{N_y} \frac{1}{L_\lambda} \left[\Sigma_{i'} \delta_{\gamma\gamma'} F_{i'} u_{i'}^{(\lambda)}\right] \left[\Sigma_i u_i^{(\lambda)} F_i^{-1} B_{i\nu}\right] \frac{\Phi_{\text{rel}} Q_\nu}{Q_\gamma} \quad (14)$$

where L_λ is the λ-th eigenvalue; $B_{i\nu}$ is a generalized microcanonical bimolecular flux coefficient for the bimolecular pair ν to form unimolecular species γ with internal energy E_η; Q_γ is the electronic-vibrational-rotational partition function of the isomer γ; Q_ν is the product of the electronic-vibrational-rotational partition functions of the members of the bimolecular pair ν; Φ_{rel} is the relative translational partition function per unit volume; and S is the number of isomers.
5) For the merged eigenmode corresponding to isomer γ', the program will find the maximum $\kappa_{\gamma',\nu}$ among bimolecular pairs ν =1, 2, .., *m*. Let it be ν'. Then the isomer γ' and bimolecular pair ν' will be recognized as in equilibrium.

This part is done in the *me_lib.py* file.

Note that the value of 0.5 in step 2 has nothing to do with **MERGETHRESHOLD**; **MERGETHRESHOLD** is EPSRS, whereas 0.5 is the normalized EPCS. We emphasize that EPSRS is used to judge if an CSE eigenvector is merged, and the normalized EPCS is used to determine which two species the merged eigenvector is describing. The threshold 0.5 of the normalized EPCS in TUMME cannot be changed by users. **MERGETHRESHOLD** should be specified by the user.

6.5 High-precision dynamic libraries

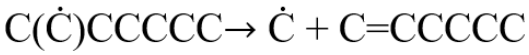
In some cases where the magnitudes of eigenvalues span a very wide range of values, it is necessary to use quadruple or octuple precision to obtain full accuracy. TUMME includes the quadruple and octuple precision code in C++. The quadruple precision version is in *dd_lib.py* file, and the octuple precision version is in *qd_lib.py* file. TUMME compiles the higher-precision code into dynamic libraries to let Python call them. We used *numpy.ctypes* as an interface between Python and C++. In the C++ code, all relaxation kernels of the isomers are re-calculated in high precision, and the construction the symmetric transition matrix **G** and its diagonalize are carried out in high precision. Finally, the C++ libraries will return eigenpairs of the symmetric transition matrix to the Python process to continue the analysis.

Users who wish to use quadruple precision or octuple precision to diagonalize their own transition matrix can use our simplified *mpack* library. First go into *mpack_dd/src* and *mpack_qd/src* and modify the include path and library path of the *qd/* folder in the *Makefile* and compile them. If it succeeds, you will get two dynamic library files called *libmpack_dd.so* and *libmpack_qd* under *mpack_dd/lib* and *mpack_qd/lib*, respectively. Then you can call these dynamic libraries by C++ under the GNU compiler. And remember to add the path of the dynamic library to your system library-path or compile your own code in

which you specify the `-Wl,-rpath` option. There are two examples, the *mpack_dd/example/eigenvector_dd.cpp* and *mpack_dd/example/eigenvector_dd.cpp*, to show how to call the *mpack* to diagonalize a matrix. The *C++/qd* folder is necessary for both libraries and should be pre-installed by using the GNU compiler. If you use the simplified *mpack*, please cite the references of *mpack*[31] and *qd*[32] in your work.

7. Test Runs

7.1 2-methylhexyl radical single-channel unimolecular dissociation

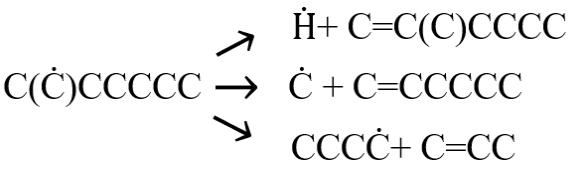


Note that hydrogens are not shown, and parentheses are used to show branching.

Figure 4. Reaction for the first test case

Anharmonicity:	SSHO
Variational effect:	None (TST)
Tunneling:	Eckart
Parallelism:	MPI with 4 processors
Precision:	Double
Standard input file:	*param.in*
Extra input files:	None
Output files:	*param.out, param.rate, kE.out, evec.out, eval.out*
Location:	*example/2MH/Eckart/*
Run command:	*tumme param.in*

7.2 2-methylhexyl radical multi-channel unimolecular dissociation



Note that hydrogens are not shown, and parentheses are used to show branching.

Figure 5. Reactions for the second test case.

Anharmonicity:	SSHO
Variational effect:	None (TST)
Tunneling:	No tunneling
Parallelism:	Serial
Precision:	Quadruple
Standard input file:	*param.in*
Extra input files:	None
Output files:	*param.out, param.rate, kE.out, evec.out, eval.out, time.txt*
Location:	*example/2MH/highprecision/*
Run command:	*tumme param.in*

7.3 Toluene + OH radical *ipso*- site addition reaction

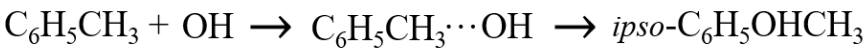


Figure 6. Reaction for the third test case

Anharmonicity:	MS for $C_6H_5CH_3$; SSHO for other species.
Variational effect:	None (TST)
Tunneling:	No tunneling
Parallelism:	MPI with 13 processors
Precision:	Double
Standard input file:	*param.in*
Extra input files:	*R_m.out*
Output files:	*param.out, param.rate, evec.out, eval.out*
Location:	*example/T+OH/ipso/*
Run command:	*tumme param.in*

7.4 Toluene + OH radical addition reactions: TST

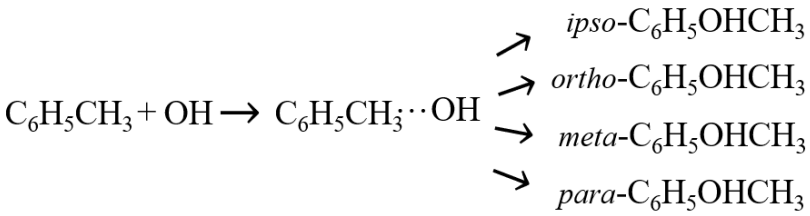


Figure 7. Reactions for the fourth test case

Anharmonicity:	SSHO
Variational effect:	None (TST)
Tunneling:	No tunneling
Parallelism:	MP with 6 processors
Precision:	Double
Standard input file:	*param.in*
Extra input files:	None
Output files:	*param.out, param.rate*
Location:	*example/T+OH/all_plain/*
Run command:	*tumme param.in*

7.5 Toluene + OH radical addition reactions: VTST

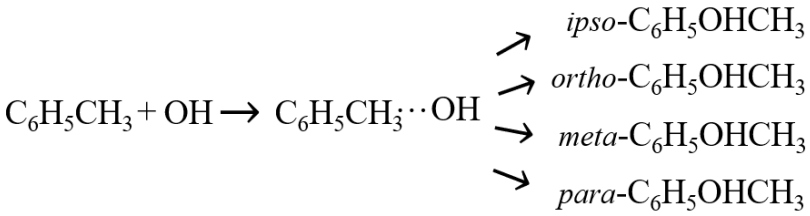


Figure 7. Reactions for fifth test case.

Anharmonicity:	MS for all species except OH
Variational effect:	CVT for four $C_6H_5CH_3$···OH association reaction channels.
Tunneling:	SCT for four $C_6H_5CH_3$···OH association reaction channels.
Parallelism:	MPI with 4 processors
Precision:	Double
Standard input file:	*param.in*
Extra input files:	*P_add_ortho.out, P_add_ipso.out, P_add_meta.out, P_add_para.out, TS_add_ortho.out, TS_add_ipso.out, TS_add_meta.out, TS_add_para.out, R_m.out Rxn2_ipso.fu6, Rxn2_para.fu6, Rxn2_meta.fu6, Rxn2_ortho.fu6*
Output files:	*param.out, param.rate, Q.txt, kE.txt*
Location:	*example/T+OH/all_calibrated/*
Run command:	*tumme param.in*

7.6 H_2CO + OH radical abstraction reaction

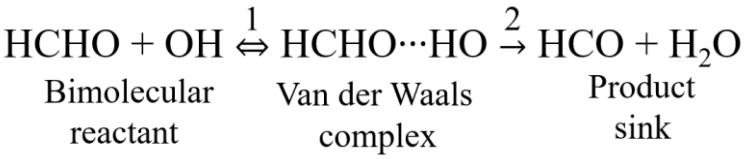


Figure 8. Reactions for sixth test case.

Anharmonicity:	MS for van der Waals complex and the transition state of reaction 2
Variational effect:	CVT for reaction 2
Tunneling:	SCT for reaction 2
Parallelism:	MPI with 11 processors
Precision:	Quadruple
Standard input file:	*param.in*

Extra input files:	mstor_complex.out, mstor_TS.out, Rxn2.fu6
Output files:	*param.out, param.rate, eval.out, time.txt*
Location:	*example/H2CO+OH/*
Run command:	*tumme param.in*

8. Bibliography of TUMME research articles

"Energy Dependence of Ensemble-Averaged Energy Transfer Moments and its Effect on Competing Decomposition Reactions," R. M. Zhang, X. Xu, and D. G. Truhlar, Journal of Physical Chemistry A **125**, 6303-6313 (2021). doi.org/10.1021/acs.jpca.1c03845
(Special Virtual Issue entitled "125 Years of The Journal of Physical Chemistry")

"TUMME: Tsinghua University Minnesota Master Equation program," R. M. Zhang, X. Xu, and D. G. Truhlar, Computer Physics Communications **270**, 108140/1-17 (2021). doi.org/10.1016/j.cpc.2021.108140

"Master Equation Study of Hydrogen Abstraction from HCHO by OH Via a Chemically Activated Intermediate," R. M. Zhang, W. Chen, D. G. Truhlar and X. Xu, Faraday Discussions 2022, online as Accepted Manuscript. doi.org/10.1039/D2FD00024e

9. Version History of TUMME

Date	Log	Version No.
12/17/2020	The first version of TUMME.	1.0
06/18/2021	• Added the dependence of $\langle \Delta E_d \rangle$ on bath temperature and on initial isomer internal energy. • Changed subkeywords from **TIMEMAX** to **TIMENUM**. • Added print option for the relaxation kernel matrix. • Improved output. • Fixed bugs.	2.0
08/05/2021	• Fixed some bugs.	2.1
08/25/2021	• Changed **EEOT** from "relative to the max 0K energy of transition states" to "relative to the max 0K energy of all species including reactants, transition states, and products" • Made it available for a reaction system without any transition state. • Fixed some bugs for reading standard files and *Polyrate* output files.	2.2
7/03/2022	• Added pseudo-first order assumption for bimolecular pairs in time-evolution, making this also available for high-precision. • Added inverse-Laplace-transform in **REACTION** block. • Abandoned rounding the size of energy bin into an integer wavenumber (**ESOT**). • Changed the way to run TUMME, provide an executable bash script. • Slightly adjusted the strategy for merger to deal with some unexpectable cases. • Adjusted the output for partition function. • Fixed some bugs.	3.0

10. References

1 S. J. Klippenstein, V. Pande, and D. G. Truhlar, Chemical Kinetics and Mechanisms of Complex Systems: A Perspective on Recent Theoretical Advances, *J. Am. Chem. Soc.* **2014**, *136*, 528-546. doi.org/10.1021/ja408723a

2 B. Widom, Molecular Transitions and Chemical Reaction Rates. *Science* **1965**, *148*, 1555−1560. doi.org/10.1126/science.148.3677.1555

3 O. K. Rice and H. C. Ramsperger, Theories of Unimolecular Gas Reactions at Low Pressures. *J. Am. Chem. Soc.* **1927**, *49*, 1617-1629. doi.org/10.1021/ja01406a001

4 L. S. Kassel, Studies in Homogeneous Gas Reactions I. *J. Phys. Chem.* **1928**, *32*, 225-242. doi.org/10.1021/j150284a007

5 Y. Georgievskii, J. A. Miller, M. P. Burke, and S. J. Klippenstein, Reformulation and Solution of the Master Equation for Multiple-Well Chemical Reactions. *J. Phys. Chem. A* **2013**, *117*, 12146−12154. doi.org/10.1021/jp4060704

6 J. Zheng, S. L. Mielke, J. L. Bao, R. Meana-Pañeda, K. L. Clarkson, and D. G. Truhlar, *MSTor* computer program – version 2017, University of Minnesota, Minneapolis, MN, 2017. https://comp.chem.umn.edu/mstor/

7 J. Zheng, S. L Mielke, K. L Clarkson, and D. G. Truhlar, MSTor: A program for calculating partition functions, free energies, enthalpies, entropies, and heat capacities of complex molecules including torsional anharmonicity. Computer Physics Communications **183**, 1803-1812 (2012). doi.org/10.1016/j.cpc.2012.03.007

8 J. Zheng, R. Meana-Pañeda, and D. G. Truhlar, *MSTor* version 2013*:* A new version of the computer code for the multistructural torsional anharmonicity with a coupled torsional potential. Computer Physics Communications **184**, 2032-2033 (2013). doi.org/10.1016/j.cpc.2013.03.011

9 (a) M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V. Ortiz, A. F. Izmaylov, J. L. Sonnenberg, D. Williams-Young, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, T. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman, and D. J. Fox, *Gaussian 09*, Gaussian, Inc., Wallingford CT, 2009.

(b) M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. V. Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V. Ortiz, A. F. Izmaylov, J. L. Sonnenberg, D. Williams-Young, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K.

Raghavachari, A. P. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman, and D. J. Fox, *Gaussian 16*, Gaussian, Inc., Wallingford CT, 2016.

10 J. Zheng, J. L. Bao, R. Meana-Pañeda, S. Zhang, B. J. Lynch, J. C. Corchado, Y.-Y. Chuang, P. L. Fast, W.-P. Hu, Y.-P. Liu, G. C. Lynch, K. A. Nguyen, C. F. Jackels, A. Fernandez Ramos, B. A. Ellingson, V. S. Melissas, J. Villà, I. Rossi, E. L. Coitiño, J. Pu, T. V. Albu, A. Ratkiewicz, R. Steckler, B. C. Garrett, A. D. Isaacson, and D. G. Truhlar, Polyrate – version 2017-C (University of Minnesota, Minneapolis, MN, 2017). https://comp.chem.umn.edu/polyrate/

11 J. Zheng; J. L. Bao, S. Zhang, J. C. Corchado, R. Meana-Pañeda, Y.-Y. Chuang, E. L. Coitiño, B. A. Ellingson, and D. G. Truhlar, *Gaussrate 17*; University of Minnesota: Minneapolis, 2017. https://comp.chem.umn.edu/gaussrate/

12 L. G. Gao, J. Zheng; J. Zheng, M. A. Iron, B. A. Ellingson, J. C. Corchado, Y.-Y. Chuang, and D. G. Truhlar, *NWChemrate 2019*; University of Minnesota: Minneapolis, 2019. https://comp.chem.umn.edu/nwchemrate/

13 H. Eyring, The Activated Complex in Chemical Reactions, *J. Chem. Phys.* **1935**, *3*, 107-115. doi.org/10.1063/1.1749604

14 J. Zheng, T. Yu, E. Papajak, I. M.Alecu, S. L. Mielke, and D. G. Truhlar, Practical Method For Including Torsional Anharmonicity in Thermochemical Calculation on Complex Molecules: The Internal-Coordinate Multi-structural Approximation. **2011**, *13*, 10885-10907. doi.org/10.1039/C0CP02644A

15 T. Yu, J. Zheng, and D. G. Truhlar, Multi-Structural Variational Transition State Theory. Kinetics of the 1,4-Hydrogen Shift Isomerization of the Pentyl Radical with Torsional Anharmonicity, *Chem. Sci.* **2011**, *2*, 2199-2213. doi.org/10.1039/C1SC00225B

16 J. Zheng and D. G. Truhlar, Including Torsional Anharmonicity in Canonical and Microcanonical Reaction Path Calculations, *J. Chem. Theory Comput.* **2013**, *9*, 2875-2881. doi.org/10.1021/ct400231q

17 B. C. Garrett and D. G. Truhlar, Criterion of Minimum State Density in the Transition State Theory of Bimolecular Reactions, *J. Chem. Phys.* **1979**, *70*, 1593-1598. doi.org/10.1063/1.437698

18 A. D. Isaacson and D. G. Truhlar, Polyatomic Canonical Variational Theory for Chemical Reaction Rates. Separable-Mode Formalism with Application to OH + H_2 $\rightarrow$ H_2O + H, *J. Chem. Phys.*, **1982**, *76*, 1380-1391. doi.org/10.1063/1.443130

19 Generalized Transition State Theory. Quantum Effects for Collinear Reactions of Hydrogen Molecules," B. C. Garrett and D. G. Truhlar, *J. Phys. Chem.* **1979***, 83*, 1079-1112 (1979). doi.org/10.1021/j100471a032

20 A. D. Isaacson, M. T. Sund, S. N. Rai, and D. G. Truhlar, Improved Canonical and Microcanonical Variational Transition State Theory Calculations for a Polyatomic System: OH + H_2 $\rightarrow$ H_2O + H, *J. Chem. Phys.* , **1985**, *82*, 1338-1340. doi.org/10.1063/1.448963

21 D. G. Truhlar and A. Kuppermann, Exact Tunneling Calculations, *J. Am. Chem. Soc.*, **1971**, *93*, 1840-1851. doi.org/10.1021/ja00737a002

22 D. G. Truhlar and A. Kuppermann, A Test of Transition State Theory Against Exact Quantum Mechanical Calculations, *Chem. Phys. Lett.* **1971**, *9*, 269-272. doi.org/10.1016/0009-2614(71)85049-2

23 B. C. Garrett, D. G. Truhlar, R. S. Grev, and A. W. Magnuson, Improved Treatment of Threshold Contributions in Variational Transition State Theory, *J. Phys. Chem.* **1980**, *84*, 1730-1748. doi.org/10.1021/j100450a013

24 Y.-P. Liu, G. C. Lynch, T. N. Truong, D.-h. Lu, D. G. Truhlar, and B. C. Garrett, Molecular Modeling of the Kinetic Isotope Effect for the [1,5]-Sigmatropic Rearrangement of *cis*-1,3-Pentadiene, *J. Am. Chem. Soc.* **1993**, *115*, 2408-2415. doi.org/10.1021/ja00059a041

25 R. M. Zhang, X. Xu, and D. G. Truhlar, "TUMME: Tsinghua University Minnesota Master Equation program," *Computer Phys. Comm. 2022, 270*, 108140. doi.org/10.1016/j.cpc.2021.108140

26 J. Troe, Theory of Thermal Unimolecular Reactions at Low Pressures. II. Strong Collision Rate Constants. Applications. *J. Chem. Phys.* **1977***, 66*, 4758-4775. doi.org/10.1063/1.433838

27 R. M. Zhang, W. Chen, D. G. Truhlar, and X. Xu, Master Equation Study of Hydrogen Abstraction from HCHO by OH Via a Chemically Activated Intermediate. *Faraday Discuss.* 2022*,* in press. doi.org/10.1039/D2FD00024e

28 H. S. Johnston and J. Heicklen, Tunneling Corrections For Unsymmetrical Eckart Potential Energy Barrier *J. Phys. Chem.* **1962**, *66*, 532-533. doi.org/10.1021/j100809a040

29 J. L. Bao, R. Meana-Pañeda, D. G. Truhlar, Multi-Path Variational Transition State Theory for Chiral Molecules: The Site-Dependent Kinetics for Abstraction of Hydrogen from Hydroperoxyl Radical, Analysis of Hydrogen Bonding in the Transition State, and Dramatic Temperature Dependence of the Activation Energy. *Chem. Sci.* **2015***, 6*, 5866-5881. doi.org/10.1039/C5SC01848J

30 I. M. Alecu, J. Zheng, Y. Zhao, and D. G. Truhlar, Computational Thermochemistry: Scale Factor Databases and Scale Factors for Vibrational Frequencies Obtained from Electronic Model Chemistries. *J. Chem. Theory Comput.* **2010**, *6*, 2872-2887. doi.org/10.1021/ct100326h

31 M. Nakata, *The MPACK (MBLAS/MLAPACK); A Multiple Precision Arithmetic Version of BLAS and LAPACK*, 2010, Version 0.6.7, http://mplapack.sourceforge.net/

32 Y. Hida, X. S. Li and D. H. Bailey, Library for Double-Double and Quad-Double Arithmetic, *NERSC Division Lawrence Berkeley National Laboratory*, 2007.