Manual
MCSI – version 2010-1

Osanna Tishchenko, Masahiro Higashi, Titus V. Albu, Jose C. Corchado,
Yongho Kim, Jordi Vill, Jianhua Xing, Hai Lin, and Donald G. Truhlar
Department of Chemistry and Supercomputing Institute,
University of Minnesota, Minneapolis, MN 55455-0431

MCSI is a computer program for generating full-dimensional potential energy surfaces for polyatomic reactions for subsequent dynamics calculations (classical trajectories, full quantum dynamics, or variational transition state theory with multidimensional tunneling).

Program version: 2010-1
Program version date: October 30, 2010
Manual version date: October 30, 2010

Note: Please consider the environment before printing this document. Please use recycled paper if printing is necessary.

CONTENTS

I. REQUIRED REFERENCES

Publications based on results obtained with this computer code should include the following references:

1. O. Tishchenko, M. Higashi, T. V. Albu, J. C. Corchado, Y. Kim, J. Vill, J. Xing, H. Lin, and D. G. Truhlar, MCSI–version 2010-1 University of Minnesota, Minneapolis, MN, 2010.

2. J. W. Ponder, TINKER–version 3.5, Washington University, St. Louis, MO, 1997.

For example: "The calculations were carried out using the MCSI[1] multi-configuration Shepard interpolation computer program, which uses single-configuration molecular mechanics subroutines from the TINKER[2] computer program."

The references are required; the wording is optional.

II. INTRODUCTION

The Multi-Configuration Shepard Interpolation (MCSI) method[3–8] is mainly based on the combination of five computational techniques: the semiempirical valence bond method,[9–22] Chang, Minichino, the Miller's method of locally estimating V_{12} for semiempirical valence bond calculations,[22,23], the Shepard interpolation method.[26,27], the use of redundant internal coordinates,[24,25] and the use of molecular mechanics in parametrizing the valence bond calcuations. The methods of Coulson and Danielsson[12] and Raff[17] and the empirical valence bond (EVB) method, which is an approach to semiempirical valence bond theory that is widely used by Warshel and coworkers,[19–21] may also be considered to be molecular-mechanics-based ways to parameterize valence bond theory, as alternatives to the quantum mechanical based approaches of Eyring, Polanyi, Sato, and Ellison.[11,14–18] Another relevant reference is the work of Downing and Michl,[32] who pointed out the possibility that the matrix elements of a nondiagonal Hamiltonian representation of the potential energy surfaces might be more amenable to approximation by simple functional forms than are the eigenvalues (which are the adiabatic potential energy surfaces). A detailed description of the MCSI method is given in Refs. 3 and 7, where the method is called multi-configuration molecular mechanics (MCMM), and the equations are given summarized in Secs. III and IV. Therefore, we present in Sec II A only a brief overview. A detailed description of the EE-MCSI method is given Ref. 6.

We now refer to the method as MCSI to emphasize that is a Shepard interpolation based on a multi-configurational framework; whereas previously we called it MCMM to emphasize that it allows calculations of potential energy surfaces that may be evaluated with a cost comparable to molecular mechanics even when the system has more than one configuration, that is, reactants and products of a chemical reaction.

The MCSI program is a composite of the `mct` module, which is a module for multi-configuration Shepard calculations, and the computer program TINKER, which is a program for single-configuration molecular mechanics calculations. The MCSI program can carry out four types of calculations: calculations based on single-configuration molecular mechanics (MM or, for emphasis, SCMM), as in TINKER itself, or calculations based on multi-configuration molecular mechanics (MCSI), calculations based on electrostatically embedded MM or SCMM (which may be called either EE-MM or EE-SCMM) in which MM is applied to systems in the presence of an electrostatic potential, and calculations based on electrostatically embedded MCSI (EE-MCSI) in which MCSI is applied to systems in the presence of an electrostatic potential. (Throughout this manual, we use MM and SCMM interchangeably as synonyms.)

Note that prior to the present version, the MCSI code has been called MC-TINKER.

A. Short description of the multi-configuration molecular mechanics method

In MCSI, we represent a nonstationary point on a potential energy surface by using two configurations. The current version of MCSI is restricted to the case of two configurations, and the rest of this manual is also restricted to the case of two configurations. For a chemical reaction, these two configurations correspond to the reactant and product configurations of that reaction. The energy at a

given point on the potential energy surface is approximated by the lowest root V of the equation

$$\begin{vmatrix} V_{11} - V & \beta \\ \beta & V_{22} - V \end{vmatrix} = 0, \quad (1)$$

where V_{11} and V_{22} are the energy functions of the two configurations, and β is the approximation to the the off-diagonal matrix element, V_{12}, also called the resonance energy function or the resonance interaction.

The V_{11} and V_{22} functions are approximated by standard or user-specified molecular mechanics functions; the standard molecular mechanics functions can be used either with standard molecular mechanics parameters or with reaction-specific parameters. The choice of β is less obvious. Several approaches for determining β are available in MCSI; the most powerful of them uses a multi-point extension of a variant of the method described in Refs. 19-20. The key element in the method of Refs. 19-20 is that β is chosen locally to reproduce a Taylor series approximation to V at one point. Here we apply that formula at one or more points called Hessian points, or electronic structure points (or electronic structure Shepard points), and we assume that V_{12} and its gradient and Hessian are zero at two other points, called the SCMM points. The Hessian points and SCMM points together are called Shepard points, and β is obtained at other points by using Shepard interpolation to join several such local approximations to provide a semiglobal approximation. From this representation, MCSI can calculate V and the analytic gradient and Hessian of V at any geometry. In summary, for the Shepard interpolation of V_{12}, the MCSI method uses a variable number of electronic structure Shepard points and up to two single-configuration molecular mechanics points, the latter two points being called the SCMM point for configuration 1 and the SCMM point for configuration 2.

Two different formalisms are available in the MCSI code. In the original version of the MCSI method,[3,4,7] the valence bond configuration matrix $\mathbf{V}$ was assumed to be Hermitian, but in the current version of the method,[5] the matrix $\mathbf{V}$ is allowed to be non-Hermitian in order to broaden the range of geometries for which the potential energy surface can be fit accurately. Another important distinction in these two formalisms is that the former one employs Shepard interpolation of modified Taylor series of V_{12}, whereas the latter uses Shepard interpolation of unmodified Taylor series of V_{12}^2 and has simpler algebra.[5] The formalism described in Ref. 5 is recommended for most cases and it is the default option in MCSI-2009.

While there are no restrictions on the locations of the electronic structure Shepard points, it is expected that these points will be placed in regions where neither of the molecular mechanics potentials are accurate. At the two SCMM points we may, as one option, define V_{12} (as well as its first and second derivatives with respect to coordinates) to be zero so the potential V reduces to V_{11} or V_{22}, whichever is smaller in energy. These points should be therefore chosen in regions where the configuration 1 and configuration 2 molecular mechanics potential functions, respectively, are accurate. However, this assumption sometimes makes the potential energy surface rough around the SCMM points because the potential energy surface calculated by the SCMM approximation is different from that calculated by the reference electronic structure method. Therefore, although it is possible to mitigate the problem by adjusting the SCMM parameters to reproduce the reference potential energy surface around the SCMM points, it may be inappropriate to employ the assumption that V_{12} is zero at certain points when one attempts to describe the potential energy surface around the reactant and product as well as around the transition state. In the current version of the MCSI program, there is an option to specify whether the two SCMM points are included in the Shepard interpolation. (See the ISHMM option in Sec. IX E 4.) Actually the user has a third option, namely to include these points, but with electronic structure data so that V_{12} is not zero. The program treats this third choice as a special case of not including SCMM points.

The Taylor series expansion of V_{12} is used to interpolate between the electronic structure theory points to calculate V_{12} for any geometry of interest on the potential energy surface. To make the approach applicable for any number of electronic structure theory points, we use a Shepard interpolation method. The Shepard scheme ensures that the potential energy surface is a continuous function that exactly reproduces the energies, gradients, and Hessians at the electronic structure theory Shepard points. The procedure allows the results to be improved by adding more electronic structure theory points, eventually converging (in principle and with sufficient care) to the potential function that would be obtained from a dense set of electronic structure theory results.

Given the geometry $\mathbf{x}$ of a point in Cartesians, for which we want to obtain the MCSI energy, gradient, and Hessian, the algorithm carries out the following steps:

- Transformation of the geometries, gradients, and Hessians of the Shepard points from Cartesian to redundant or nonredundant internal coordinates $\mathbf{r}$ (see sec. II B for the explanation of the internal coordinates used in the MCSI code).

- Calculation of V_{12} and its first and second derivatives in internal coordinates at all geometries for which electronic structure theory gradient and Hessian data are available.

- Transformation of an input geometry $\mathbf{x}$ in Cartesians to the same set of redundant or nonredundant internal coordinates $\mathbf{r}(\mathbf{x})$.

- Estimation of V_{12} and its first and second derivatives with respect to internal coordinates at the geometry $\mathbf{r}(\mathbf{x})$ by means of Shepard interpolation, which involves a weight function written in terms of coordinates $\mathbf{s}(\mathbf{x})$.

- Transformation of the internal-coordinate derivatives of V_{12} at $\mathbf{r}(\mathbf{x})$ to Cartesian coordinates by the inverse of the transformation of a previous step (note that using the inverse transformation preserves the correct orientation in Cartesian coordinates).

- Calculation of V_{11} and V_{22} and their derivatives at the geometry $\mathbf{x}$ in Cartesian coordinates (TINKER evaluates the derivatives in internal coordinates $\mathbf{q}$ and then transforms them to Cartesian coordinates).
- Calculation of $V(\mathbf{x})$ and its first and second derivatives in Cartesian coordinates.

B. Usage of the internal coordinates in MCSI

The MCSI code returns the gradients and Hessians in Cartesian coorinates. However, in order to make the method rotationally and translationally invariant, the interpolation procedure is carried out in internal rather than in Cartesian coordinates, in particular either in redundant or nonredundant internal coordinates.

The internal coordinates appear in the MCSI method in three different contexts:

- the set $\mathbf{q}$ used to evaluate MM energies and derivatives,
- the set $\mathbf{r}$ used for the Shepard interpolation step, and
- the set $\mathbf{s}$ used to evaluate the weight function.

The distinction between these coordinates was not explicitly mentioned in Refs. 3-31, but it is emphasized in Refs. 4 and 7 and in Secs. III and IV of this manual. In Refs. 3-31, the notation $\mathbf{q}$ was used for all three sets. While the composition of the set $\mathbf{q}$ is completely determined by the molecular mechanics method, sets $\mathbf{r}$ and $\mathbf{s}$ are introduced in MCSI. Both sets $\mathbf{r}$ and $\mathbf{s}$ should be specified by the user in the input `esp.fu85` file. The current version of the code only supports the following cases:

- sets $\mathbf{r}$ and $\mathbf{s}$ are the same
- set $\mathbf{s}$ is a subset of $\mathbf{r}$
- a special case when set $\mathbf{s}$ is different from set $\mathbf{r}$; this case is designed for a generic BX$\cdots$A$\rightarrow$B$\cdots$XA atom transfer reaction and it is described in detail below.

Other cases require additional considerations.

1. *Special case of the internal coordinates for an atom transfer reaction*

This is the only case currently available in the MCSI code in which set $\mathbf{s}$ includes internal coordinates that are not present in set $\mathbf{r}$.

Let B–X$\cdots$A$\rightarrow$B$\cdots$XA be an atom transfer reaction. The coordinates $\mathbf{r}$ and $\mathbf{s}$ are selected as follows:

- set $\mathbf{s}$: three interatomic distances $\{r_{BX}; r_{AX}; r_{AB}\}$
- set $\mathbf{r}$: either nonredundant internal coordinates or redundant internal coordinates that involve the following three coordinates $\{r_{BX}; r_{AX}; \angle_{BXA}\}$, i.e., two bond distances, r_{BX} and r_{AX}, and the angle, $\angle_{BXA}$, as a subset.

An example is shown in Ref. 5, Appendix B. Other choices of $\mathbf{s}$ and $\mathbf{r}$ require additional programming.

C. Handling of nuclear permutation symmetry

The current version of the code can generate semi-global potential energy surfaces that are invariant with respect to permutations of selected identical nuclei. The current version supports cases when only three identical nuclei need to be treated as symmetrically equivalent while the rest (if any) can be assumed to be distinguishable. These cases are labeled $m = 3$. Extensions to other cases ($m = 2$ or $m \geq 3$) are in principle straightforward and can be implemented in future versions. This manual describes the input to both symmetrized and non-symmetrized calculations with MCSI. For details of the permutation symmetry algorithm see Ref. 4 and the Appendix to this manual. We note that in symmetrized calculations with $m \geq 3$ we use the redundant-internal-coordinate subroutines even if the number of the coordinates is equal $3N$-6 (e.g. when one uses all internuclear distances for OH + HH system) because in this case the coordinates are not internally independent.

D. Zero of energy in the multi-configuration Shepard interpolation method

Special attention should be paid to the zero of energy in an MCSI calculation. The two molecular mechanics (MM) force field and the electronic structure (ES) calculation each have their own origin for the zero of energy. The MM energy of a molecule is the sum of several terms. For some small molecules like OH, CH_4, H_2O, and CH_3, the MM energies for the optimized structures are zero. Thus, for example, the MM energy of reaction for the OH + CH_4 $\rightarrow$ H_2O + CH_3 reaction is zero, because both the MM reactant state energy of OH + CH_4 and the MM product state energy of H_2O + CH_3 are zero. This does not mean that the actual energy of reaction is zero, though, because one cannot compare the MM energies for different molecules that have different atom types and connectivities, since they are from different MM potential functions. For more complicated molecules, such as propane, the MM energy is not necessarily zero even in the optimized structure. (In general, whether or not the MM energy is zero at the equilibrium structure depends on which MM force field one is considering.)

To connect the MM energies for the two configurations with the ES energies, a unique scale of energies should be defined. This energy scale is defined in the MCSI calculations based on the MM and ES energies at some states (or geometries) that we call energy reference states. The program runs if the energy reference states are chosen at any

arbitrary state or geometry. We alert the user though that it is more physical to choose the energy reference states where the molecular mechanics potentials are relatively accurate, i.e., close to equilibrium structures. (Note that the energy reference states do not need to be the same as the SCMM points, which are the single-configuration molecular mechanics points used in Shepard interpolation.)

The MM evaluations of the V_{11} and V_{22} matrix elements at the energy reference states (obtained using TINKER) have the following values:

$$V_{11}(\text{configuration 1 energy reference state}) = \texttt{vz1}^{\text{MM}} \quad (2)$$

$$V_{22}(\text{configuration 2 energy reference state}) = \texttt{vz2}^{\text{MM}} \quad (3)$$

For emphasis we use superscript MM to denote values that are obtained through molecular mechanics calculations. As stated above, $\texttt{vz1}^{\text{MM}}$ and $\texttt{vz2}^{\text{MM}}$ may be both zero for some simple cases if the energy reference states are the optimized structures but in the general case may have any value.

The electronic structure theory energies, which are the high-level energies from *Gaussian* or other electronic structure packages, satisfy:

$$V(\text{configuration 1 energy reference state}) = \texttt{Vz1}^{\text{ES}} \quad (4)$$

$$V(\text{configuration 2 energy reference state}) = \texttt{Vz2}^{\text{ES}} \quad (5)$$

We need to have this energy scale consistent with the one obtained from MM calculations. We define the zero of energy to be the electronic structure theory energy at the configuration 1 energy reference state. Therefore:

$$V(\text{configuration 1 energy reference state}) = 0 \quad (6)$$

$$V(\text{configuration 2 energy reference state}) = \texttt{ediff0} \quad (7)$$

where `ediff0` is given by:

$$\texttt{ediff0} = \texttt{Vz2}^{\text{ES}} - \texttt{Vz1}^{\text{ES}} \quad (8)$$

and it is obtained only from electronic structure calculations. Then, whenever MCSI determines a MM energy, V_{11} or V_{22}, these energies will be adjusted as:

$$V_{11}(\text{mcsi}) = V_{11}(\text{tinker}) - \texttt{vz1}^{\text{MM}} \quad (9)$$

$$V_{22}(\text{mcsi}) = V_{22}(\text{tinker}) - \texttt{vz2}^{\text{MM}} + \texttt{ediff0} \quad (10)$$

Furthermore, when an electronic structure energy V is used in an MCSI calculation, we use the adjusted energy value given by:

$$V(\text{mcsi}) = V^{\text{ES}} - \texttt{Vz1}^{\text{ES}} \quad (11)$$

Later in this manual, we sometimes use another notation, in particular, $\texttt{vz1}^{\text{MM}}$ is called ZERO1, $\texttt{vz2}^{\text{MM}}$ is called ZERO2, and `ediff0` is called EDIFF (that is, ZERO1, ZERO2, and EDIFF are the keywords whose arguments are $\texttt{vz1}^{\text{MM}}$, $\texttt{vz2}^{\text{MM}}$, and `ediff0`).

E. Short description of the electrostatically embedded molecular mechanics (EE-MM) method

EE-MM is a combined quantum mechanical and molecular mechanical (QM/MM) method, based on Refs. 33 and 34, that can reproduce the electrostatically embedded QM energy $V^{\text{EEQM}}(\mathbf{x}, \mathbf{\Phi})$ around a certain point $(\mathbf{x}_0, \mathbf{\Phi}_0)$; here $V^{\text{EEQM}}(\mathbf{x}, \mathbf{\Phi})$ is the sum of the QM energy and QM/MM electrostatic interaction energy with a site-site representation,

$$V^{\text{EEQM}}(\mathbf{x}, \mathbf{\Phi}) = \langle \Psi | \hat{H}_0 + \hat{\mathbf{Q}}^{\mathsf{T}} \mathbf{\Phi} | \Psi \rangle \quad (12)$$

where $\mathbf{x}$ stands for the collection of the Cartesian coordinates $\mathbf{x}_\alpha$ ($\alpha = 1, 2, \ldots, N^{\text{QM}}$), where N^{QM} is the number of the QM atoms) in the QM region, Ψ is the electronic wave function, $\hat{H}_0$ is the electronic Hamiltonian (including nuclear repulsion) of the QM region, $\hat{\mathbf{Q}}$ is the population operator vector of order N^{QM} whose components $\hat{Q}_\alpha$ are the population operators that generate the partial charges on QM atomic sites α:

$$Q_\alpha = \langle \Psi | \hat{Q}_\alpha | \Psi \rangle \quad (13)$$

and $\mathbf{\Phi}$ is the electrostatic potential distribution, which is a vector on the order of N^{QM}, each of whose components Φ_α is the electrostatic potential at atom α from the MM region.

In order to evaluate $V^{\text{EEQM}}(\mathbf{q}, \mathbf{\Phi})$ in the vicinity of a certain point $(\mathbf{q}_0, \mathbf{\Phi}_0)$, a second-order Taylor expansion is made:

$$V^{\text{EEQM}}(\mathbf{q}, \mathbf{\Phi}) \approx V_0^{\text{EEQM}} + \frac{\partial V^{\text{EEQM}}}{\partial \mathbf{q}}^{\mathsf{T}} \Delta\mathbf{q} + \frac{1}{2} \Delta\mathbf{q}^{\mathsf{T}} \frac{\partial^2 V^{\text{EEQM}}}{\partial \mathbf{q}^2} \Delta\mathbf{q} + \frac{\partial V^{\text{EEQM}}}{\partial \mathbf{\Phi}}^{\mathsf{T}} \Delta\mathbf{\Phi} + \frac{1}{2} \Delta\mathbf{\Phi}^{\mathsf{T}} \frac{\partial^2 V^{\text{EEQM}}}{\partial \mathbf{\Phi}^2} \Delta\mathbf{\Phi} + \Delta\mathbf{\Phi}^{\mathsf{T}} \frac{\partial^2 V^{\text{EEQM}}}{\partial \mathbf{\Phi} \partial \mathbf{q}} \Delta\mathbf{q} \quad (14)$$

where $V_0^{\text{EEQM}} = V^{\text{EEQM}}(\mathbf{q}_0, \mathbf{\Phi}_0)$, $\Delta\mathbf{q} = \mathbf{q} - \mathbf{q}_0$, and $\Delta\mathbf{\Phi} = \mathbf{\Phi} - \mathbf{\Phi}_0$. Here we employ arbitrary coordinates $\mathbf{q}$; $\mathbf{q}$ can be

the Cartesian coordinates $\mathbf{x}$ or internal ones $\mathbf{r}$. The first derivative of V^{EEQM} with respect to a component of $\mathbf{\Phi}$ is given by

$$\frac{\partial V^{\mathrm{EEQM}}}{\partial \Phi_\alpha} = \langle \Psi | \hat{Q}_\alpha | \Psi \rangle = Q_\alpha \tag{15}$$

Then, the second partial derivatives of V^{EEQM} are

$$\frac{\partial^2 V^{\mathrm{EEQM}}}{\partial \Phi_\alpha \partial \Phi_\beta} = \frac{\partial Q_\alpha}{\partial \Phi_\beta} \equiv \chi_{\alpha\beta} \tag{16}$$

and

$$\frac{\partial^2 V^{\mathrm{EEQM}}}{\partial \Phi_\alpha \partial q_\beta} = \frac{\partial Q_\alpha}{\partial q_\beta} \equiv \kappa_{\alpha\beta} \tag{17}$$

These variables, $\chi_{\alpha\beta}$ and $\kappa_{\alpha\beta}$, are known as charge response kernels (CRKs) (Refs. 33 and 35); they describe the QM charge fluctuations due to the external electrostatic potential and to the displacement of the QM atoms.

There can be some choices for the evaluation of other terms,

$$V_0^{\mathrm{EEQM}} + \frac{\partial V^{\mathrm{EEQM}}}{\partial \mathbf{q}} \Delta \mathbf{q} + \frac{1}{2} \Delta \mathbf{q}^{\mathsf{T}} \frac{\partial^2 V^{\mathrm{EEQM}}}{\partial \mathbf{q}^2} \Delta \mathbf{q},$$

which depend only on $\mathbf{q}$. For example, Lu and Yang directly evaluates these terms using the Cartesian coordinates (Ref. 35). In the current version of the MCSI program, as in the case of Ref. 34, we employ the internal coordinates $\mathbf{r}$, and the terms

$$V_0^{\mathrm{EEQM}} + \frac{\partial V^{\mathrm{EEQM}}}{\partial \mathbf{r}} \Delta \mathbf{r} + \frac{1}{2} \Delta \mathbf{r}^{\mathsf{T}} \frac{\partial^2 V^{\mathrm{EEQM}}}{\partial \mathbf{r}^2} \Delta \mathbf{r},$$

which depend only on $\mathbf{r}$, are replaced with the MM potential energy function. Therefore, we define the EE-MM (EE-SCMM) potential energy function as

$$V^{\text{EE-MM}}(\mathbf{r}, \mathbf{\Phi}) = V^{\mathrm{MM}}(\mathbf{r}) + V^{\mathrm{CRK}}(\mathbf{r}, \mathbf{\Phi}) \tag{18}$$

where V^{MM} is the MM potential energy function (which is calculated by the TINKER program) and

$$V^{\mathrm{CRK}}(\mathbf{r}, \mathbf{\Phi}) = \mathbf{Q}_0^{\mathsf{T}} \Delta \mathbf{\Phi} + \frac{1}{2} \Delta \mathbf{\Phi}^{\mathsf{T}} \boldsymbol{\chi}_0 \Delta \mathbf{\Phi} + \Delta \mathbf{\Phi}^{\mathsf{T}} \boldsymbol{\kappa}_0 \Delta \mathbf{r} \tag{19}$$

where $\mathbf{Q}_0$, $\boldsymbol{\chi}_0$, and $\boldsymbol{\kappa}_0$ are the partial charges and CRKs at $(\mathbf{r}_0, \mathbf{\Phi}_0)$. Note that the last term in eq. 19 is ignored in Ref. 34. Then we calculate the EE-MM energy $V^{\text{EE-MM}}(\mathbf{r}, \mathbf{\Phi})$ around $(\mathbf{r}_0, \mathbf{\Phi}_0)$. Note that if one attempts to describe a global potential energy surface along a reaction path, one should employ the EE-MCSI method (See the next section) rather than EE-MM, which is best suited to nonreactive systems.

F. Short description of the electrostatically embedded multi-configuration molecular mechanics (EE-MCSI) method

The EE-MCSI method (Ref. 6) extends the original MCSI method so that it can be applied to systems in the presence of an electrostatic potential. Alternatively, it may be thought of as extending EE-MM to EE-MCSI. EE-MCSI is a combined quantum mechanical and molecular mechanical (QM/MM) method, and it can reproduce the electrostatically embedded QM energy V^{EEQM}, which is the sum of QM energy and QM/MM electrostatic interaction energy with a site-site representation,

$$V^{\mathrm{EEQM}}(\mathbf{x}, \mathbf{\Phi}) = \langle \Psi | \hat{H}_0 + \hat{\mathbf{Q}}^{\mathsf{T}} \mathbf{\Phi} | \Psi \rangle \tag{20}$$

where $\mathbf{x}$ stands for the collection of the Cartesian coordinates $\mathbf{x}_\alpha$ ($\alpha = 1, 2, \ldots, N^{\mathrm{QM}}$), where N^{QM} is the number of the QM atoms) in the QM region, Ψ is the electronic wave function, $\hat{H}_0$ is the electronic Hamiltonian (including nuclear repulsion) of the QM region, $\hat{\mathbf{Q}}$ is the population operator vector of order N^{QM} whose components $\hat{Q}_\alpha$ are the population operators that generate the partial charges on QM atomic sites α:

$$Q_\alpha = \langle \Psi | \hat{Q}_\alpha | \Psi \rangle \tag{21}$$

and $\mathbf{\Phi}$ is the electrostatic potential distribution, which is a vector on the order of N^{QM}, each of whose components Φ_α is the electrostatic potential at atom α from the MM region.

As in the case of the original MCSI method, the potential energy in EE-MCSI is defined as the lowest eigenvalue of a 2×2 diabatic Hamiltonian matrix,

$$\mathbf{V}^{\text{EE-MCSI}}(\mathbf{r}, \mathbf{\Phi}) = \begin{pmatrix} V_{11}(\mathbf{r}, \mathbf{\Phi}) & V_{12}(\mathbf{r}, \mathbf{\Phi}) \\ V_{12}(\mathbf{r}, \mathbf{\Phi}) & V_{22}(\mathbf{r}, \mathbf{\Phi}) \end{pmatrix}, \tag{22}$$

where we use nonredundant or redundant internal coordinates $\mathbf{r}$ to represent the nuclear coordinates of the QM subsystem. V_{11} and V_{22} are analytic functions that describe V^{EEQM} in the region of reactants and products. V_{12} is based on a set of Shepard points $(\mathbf{r}^{(k)}, \mathbf{\Phi}^{(k)})$ where $k = 1, 2, \ldots, N$. We evaluate $[V_{12}(\mathbf{r}, \mathbf{\Phi}; k)]^2$ by a second-order Taylor expansion around each Shepard point $(\mathbf{r}^{(k)}, \mathbf{\Phi}^{(k)})$ where the Taylor series coefficients are determined such that $V^{\text{EE-MCSI}}$ reproduces V^{EEQM} and its first and second derivatives with respect to $\mathbf{r}$ and $\mathbf{\Phi}$ at Shepard point $(\mathbf{r}^{(k)}, \mathbf{\Phi}^{(k)})$. Then, we construct $V_{12}(\mathbf{r}, \mathbf{\Phi})$ at any arbitrary geometry by Shepard interpolation of these expressions.

To implement the above procedure, one needs the derivatives of electronic structure calculations of $V^{\mathrm{EEQM}}(\mathbf{r}, \mathbf{\Phi})$ with respect to $\mathbf{\Phi}$ in addition to those with respect to $\mathbf{r}$. The first derivative of V^{EEQM} with respect to a component of $\mathbf{\Phi}$ is given by

$$\frac{\partial V^{\mathrm{EEQM}}}{\partial \Phi_\alpha} = \langle \Psi | \hat{Q}_\alpha | \Psi \rangle = Q_\alpha \tag{23}$$

Then, the second partial derivatives of V^{EEQM} are

$$\frac{\partial^2 V^{\mathrm{EEQM}}}{\partial\Phi_\alpha\partial\Phi_\beta} = \frac{\partial Q_\alpha}{\partial\Phi_\beta} \equiv \chi_{\alpha\beta} \tag{24}$$

and

$$\frac{\partial^2 V^{\mathrm{EEQM}}}{\partial\Phi_\alpha\partial r_\beta} = \frac{\partial Q_\alpha}{\partial r_\beta} \equiv \kappa_{\alpha\beta} \tag{25}$$

These variables, $\chi_{\alpha\beta}$ and $\kappa_{\alpha\beta}$, are known as charge response kernels (CRKs) (Refs. 33 and 35); they describe the QM charge fluctuations due to the external electrostatic potential and to the displacement of the QM atoms. Since these effects are usually not included in MM potential energy functions, we define the EE-MM (EE-SCMM) potential energy function as (see previous section)

$$V_{ii}(\mathbf{r},\mathbf{\Phi}) = V_{ii}^{\mathrm{MM}}(\mathbf{r}) + V_{ii}^{\mathrm{CRK}}(\mathbf{r},\mathbf{\Phi}) \tag{26}$$

where V_{ii}^{MM} is the MM potential energy function and

$$\begin{aligned} V_{ii}^{\mathrm{CRK}}(\mathbf{r},\mathbf{\Phi}) =& \mathbf{Q}_0^{(i)\mathsf{T}}\Delta\mathbf{\Phi} + \frac{1}{2}\Delta\mathbf{\Phi}^{(i)\mathsf{T}}\boldsymbol{\chi}_0^{(i)}\Delta\mathbf{\Phi}^{(i)} \\ &+ \Delta\mathbf{\Phi}^{(i)\mathsf{T}}\boldsymbol{\kappa}_0^{(i)}\Delta\mathbf{r}^{(i)} \end{aligned} \tag{27}$$

where $\mathbf{Q}^{(i)}$, $\boldsymbol{\chi}^{(i)}$, and $\boldsymbol{\kappa}^{(i)}$ are calculated values at the reactant and product, such that the partial charges and CRKs calculated by the EE-MCSI agree with electronic structure calculation at the reactant and product, respectively. (In Ref. 6, the reactant and product correspond to infinitely separated reagents, but the MCSI program allows them to be either infinitely separated or at a finite distance, as in a van der Waals well or an ion-dipole complex.) Then one can calculate the EE-MCSI potential energy and its derivatives.

The calculation steps for EE-MCSI are the same as those for MCSI in Secs. III and IV below except that $\mathbf{\Phi}$ is added.

III. MCSI ALGORITHM (NONSYMMETRIZED)

The procedure for constructing a potential energy surface using a nonsymmetrized MCSI algorithm involves the following steps:

(i) a) Select k = 1, 2, ..., N molecular geometries to be used as electronic structure Shepard points, and b) calculate the energies $V^{(k)}$, gradients $\mathbf{G}^{(k)}$, and Hessians $\mathbf{F}^{(k)}$ at these geometries (e.g., using the GAUSSIAN code).

(ii) Read electronic structure information (accurate energies $V^{(k)}$, gradients $\mathbf{G}^{(k)}$, and Hessians $\mathbf{F}^{(k)}$) k = 1, 2, ..., N, and molecular mechanics information ($V_{nn}^{(k)}$, $\mathbf{G}_{nn}^{(k)}$, $\mathbf{F}_{nn}^{(k)}$) for n = 1, 2 and k = 1, 2, ..., N in Cartesian coordinates. ($V_{nn}^{(k)}$, $\mathbf{G}_{nn}^{(k)}$, $\mathbf{F}_{nn}^{(k)}$ are obtained by calling the TINKER code).

(ii) Transform $\mathbf{G}^{(k)}$, $\mathbf{F}^{(k)}$, $\mathbf{G}_{nn}^{(k)}$, and $\mathbf{F}_{nn}^{(k)}$ to the set of internal coordinates $\mathbf{r}$ by the Wilson B matrix and C tensor. This yields:

$$\mathbf{g}^{(k)} \equiv \frac{\partial}{\partial\mathbf{r}} V\Big|_{\mathbf{r}=\mathbf{r}(\mathbf{x}^{(k)})} \tag{28}$$

$$\mathbf{f}^{(k)} \equiv \frac{\partial^2}{\partial\mathbf{r}^2} V\Big|_{\mathbf{r}=\mathbf{r}(\mathbf{x}^{(k)})} \tag{29}$$

$$\mathbf{g}_{nn}^{(k)} \equiv \frac{\partial}{\partial\mathbf{r}} V_{nn}\Big|_{\mathbf{r}=\mathbf{r}(\mathbf{x}^{(k)})} \tag{30}$$

$$\mathbf{f}_{nn}^{(k)} \equiv \frac{\partial^2}{\partial\mathbf{r}^2} V_{nn}\Big|_{\mathbf{r}=\mathbf{r}(\mathbf{x}^{(k)})} \tag{31}$$

In these notes, we will use capital $\mathbf{G}$ and $\mathbf{F}$ to denote the gradients and Hessians with respect to Cartesian coordinates, and lower case $\mathbf{g}$ and $\mathbf{f}$ to denote the corresponding derivatives with respect to internal coordinates.

(iv) Define a matrix $\mathbf{V}^{(k)}$ at each geometry (k) by

$$\mathbf{V}^{(k)}(\mathbf{r}) = \begin{pmatrix} V_{11}^{(k)}(\mathbf{r}) & V_{12}(\mathbf{r};k) \\ V_{12}(\mathbf{r};k) & V_{22}^{(k)}(\mathbf{r}) \end{pmatrix} \tag{32}$$

and construct second-order Taylor series expansions of $T_{12}^2 \equiv V_{12}^2$, around each data point (k) by:

$$\begin{aligned} [T_{12}(\mathbf{r};k)]^2 &\equiv [V_{12}(\mathbf{r};k)]^2 \\ &\approx \left(V_{11}^{(k)} - V^{(k)}\right)\left(V_{22}^{(k)} - V^{(k)}\right) + \left(V_{22}^{(k)} - V^{(k)}\right)\left(\mathbf{g}_{11}^{(k)} - \mathbf{g}^{(k)}\right)^{\mathsf{T}}\Delta\mathbf{r}^{(k)} \\ &\quad + \left(V_{11}^{(k)} - V^{(k)}\right)\left(\mathbf{g}_{22}^{(k)} - \mathbf{g}^{(k)}\right)^{\mathsf{T}}\Delta\mathbf{r}^{(k)} + \frac{1}{2}\left(V_{22}^{(k)} - V^{(k)}\right)\Delta\mathbf{r}^{(k)\mathsf{T}}\left(\mathbf{f}_{11}^{(k)} - \mathbf{f}^{(k)}\right)\Delta\mathbf{r}^{(k)} \\ &\quad + \frac{1}{2}\left(V_{11}^{(k)} - V^{(k)}\right)\Delta\mathbf{r}^{(k)\mathsf{T}}\left(\mathbf{f}_{22}^{(k)} - \mathbf{f}^{(k)}\right)\Delta\mathbf{r}^{(k)} + \left(\mathbf{g}_{11}^{(k)} - \mathbf{g}^{(k)}\right)^{\mathsf{T}}\Delta\mathbf{r}^{(k)}\left(\mathbf{g}_{22}^{(k)} - \mathbf{g}^{(k)}\right)^{\mathsf{T}}\Delta\mathbf{r}^{(k)} \end{aligned} \tag{33}$$

where

$$\Delta\mathbf{r}^{(k)} = \mathbf{r}(\mathbf{x}) - \mathbf{r}(\mathbf{x}^{(k)}). \tag{34}$$

This step uses the Taylor series reversion $V_{12}^2(\mathbf{x}) = [V_{11}(\mathbf{x}) - V(\mathbf{x})][V_{22}(\mathbf{x}) - V(\mathbf{x})]$ (Chang & Miller, Ref. 22).

(v) For each geometry (k) calculate Taylor coefficients

$D^{(k)}$, $\mathbf{b}^{(k)}$, and $\mathbf{C}^{(k)}$:

$$D^{(k)} = \left(V_{11}^{(k)} - V^{(k)}\right)\left(V_{22}^{(k)} - V^{(k)}\right) \tag{35}$$

$$\mathbf{b}^{(k)} = \frac{\mathbf{g}_{11}^{(k)} - \mathbf{g}^{(k)}}{V_{11}{}^{(k)} - V^{(k)}} - \frac{\mathbf{g}_{22}^{(k)} - \mathbf{g}^{(k)}}{V_{22}{}^{(k)} - V^{(k)}} \tag{36}$$

$$\begin{aligned}\mathbf{C}^{(k)} =& \frac{1}{D^{(k)}}\left[\left(\mathbf{g}_{11}^{(k)} - \mathbf{g}^{(k)}\right)\left(\mathbf{g}_{22}^{(k)} - \mathbf{g}^{(k)}\right)^{\mathsf{T}}\right. \\ &\left. + \left(\mathbf{g}_{22}^{(k)} - \mathbf{g}^{(k)}\right)\left(\mathbf{g}_{11}^{(k)} - \mathbf{g}^{(k)}\right)^{\mathsf{T}}\right] \\ &+ \frac{\mathbf{f}_{11}^{(k)} - \mathbf{f}^{(k)}}{V_{11}^{(k)} - V^{(k)}} + \frac{\mathbf{f}_{22}^{(k)} - \mathbf{f}^{(k)}}{V_{22}^{(k)} - V^{(k)}}\end{aligned} \tag{37}$$

With these coeffiecients, eq. 33 can be rewritten as:

$$[T_{12}(\mathbf{r};k)]^2 = D^{(k)}\left(1 + \mathbf{b}^{(k)\mathsf{T}}\Delta\mathbf{r}^{(k)} + \frac{1}{2}\Delta\mathbf{r}^{(k)\mathsf{T}}\mathbf{C}^{(k)}\Delta\mathbf{r}^{(k)}\right). \tag{38}$$

Note that steps (i)–(v) are performed once at the beginning. Then steps (vi) and (vii) are carried out every time that the dynamics algorithm needs the energy, gradient, and/or Hessian.

(vi) Define the valence bond configuration interaction matrix $\mathbf{V}$ at the input geometry $\mathbf{x}$ by

$$\mathbf{V}(\mathbf{x}) = \begin{pmatrix} V_{11}(\mathbf{x}) & \beta(\mathbf{x}) \\ \beta(\mathbf{x}) & V_{22}(\mathbf{x}) \end{pmatrix}, \tag{39}$$

The lowest-energy eigenvalue of this matrix is the MCSI potential energy function. The diagonal matrix elements $V_{nn}(\mathbf{x})$ and their derivatives $\mathbf{G}_{nn}(\mathbf{x})$, and $\mathbf{F}_{nn}(\mathbf{x})$ are defined by molecular mechanics. The off-diagonal matrix elements $\beta(\mathbf{x})$ are obtained via Shepard interpolation in internal coordinates $\mathbf{r}(\mathbf{x})$. Two different approaches are available in the MCSI code to calculate β. One is via Shepard interpolation of T_{12}^2 of eq. 38 as described in Ref. 5. In particular, the Shepard interpolation step yields

$$\beta_o^2(\mathbf{r}) = \sum_{k=1}^{N} w_k(\mathbf{s}) T_{12}^2(\mathbf{r};k), \tag{40}$$

where w_k are normalized weights, and each quantity $T_{12}^2(\mathbf{r};k)$ is a second-order Taylor series of $V_{12}^{(k)2}$ at a geometry $\mathbf{r}$. Then, β is approximated by

$$\beta(\mathbf{r}) = \begin{cases} |\beta_o(\mathbf{r})|; & \beta_o^2(\mathbf{r}) \geq 0 \\ iu|\beta_o(\mathbf{r})|; & \beta_o^2(\mathbf{r}) < 0, \end{cases} \tag{41}$$

where

$$u(\mathbf{r}) = \begin{cases} 1; & \beta_o^2(\mathbf{r}) \geq -\Delta^2/4 \\ \Delta/(2|\beta_o|); & \beta_o^2(\mathbf{r}) < -\Delta^2/4, \end{cases} \tag{42}$$

and

$$\Delta = V_{11}(\mathbf{r}) - V_{22}(\mathbf{r}). \tag{43}$$

The first and second derivatives of β^2 with respect to the coordinates $\mathbf{r}$ are the same as the derivatives of β_o^2 for all $\beta_o^2 \geq -\Delta^2/4$. These derivatives are given by

$$\frac{\partial \beta_o^2}{\partial \mathbf{r}} = \sum_{k=1}^{N}\left(\frac{\partial w_k}{\partial \mathbf{r}} T_{12}^2(\mathbf{r};k) + w_k \mathbf{g}_{12}(\mathbf{r},k)\right) \tag{44}$$

and

$$\begin{aligned}\frac{\partial^2 \beta_o^2}{\partial \mathbf{r}^2} =& \sum_{k=1}^{N}\left(\frac{\partial^2 w_k}{\partial \mathbf{r}^2} T_{12}^2(\mathbf{r};k) + \frac{\partial w_k}{\partial \mathbf{r}}\mathbf{g}_{12}(\mathbf{r};k)^{\mathsf{T}}\right. \\ &\left. + \mathbf{g}_{12}(\mathbf{r};k)\left(\frac{\partial w_k}{\partial \mathbf{r}}\right)^{\mathsf{T}} + w_k D^{(k)}\mathbf{C}^{(k)}\right),\end{aligned} \tag{45}$$

where

$$\mathbf{g}_{12}(\mathbf{r};k) \equiv \frac{\partial T_{12}^2(\mathbf{r};k)}{\partial \mathbf{r}} = D^{(k)}\left(\mathbf{b}^{(k)} + \mathbf{C}^{(k)}\boldsymbol{\Delta}\mathbf{r}^{(k)}\right), \tag{46}$$

$$\frac{\partial w_k}{\partial r_\alpha} = \sum_{\gamma=1}^{\Gamma}\frac{\partial w_k}{\partial s_\gamma}\frac{\partial s_\gamma}{\partial r_\alpha}, \tag{47}$$

and

$$\frac{\partial^2 w_k}{\partial r_\alpha \partial r_\beta} = \sum_{\gamma=1}^{\Gamma}\sum_{\gamma=1'}^{\Gamma}\frac{\partial s_\gamma}{\partial r_\alpha}\frac{\partial^2 w_k}{\partial s_\gamma \partial s_{\gamma'}}\frac{\partial s_{\gamma'}}{\partial r_\beta} + \sum_{\gamma=1}^{\Gamma}\frac{\partial w_k}{\partial s_\gamma}\frac{\partial^2 s_\gamma}{\partial r_\alpha \partial r_\beta}, \tag{48}$$

where $\mathbf{r}$ and $\mathbf{s}$ are the sets of the internal coordinates used in Shepard interpolation and in calculations of the weight function.

In this approach,[5] called the non-Hermitian MCSI, the matrix $\mathbf{V}$ is allowed to be non-Hermitian, and this is the default method in MCSI 2010-1.

The other approach to calculated β is via Shepard interpolation of modified T_{12}^2 around N data points as described in Ref. 3. In this case, the Shepard interpolation step yields

$$\beta(\mathbf{r}) = \sum_{k=1}^{N} w_k(\mathbf{r}) V'_{12}(\mathbf{r};k), \tag{49}$$

where w_k are normalized weights, and V'_{12} is defined by

$$V'_{12}(\mathbf{r};k) = \sqrt{T_{12}^2(\mathbf{r};k)u(\mathbf{r};k)}, \tag{50}$$

where $T_{12}^2(\mathbf{r},k)$ is given in eq. 38 , and

$$u(\mathbf{r};k) = \begin{cases} \dfrac{1}{1 + (\Delta/T_{12}(\mathbf{r};k))^{2n}}; & T_{12}^2(\mathbf{r};k) > 0 \\ 0; & \text{otherwise.} \end{cases} \tag{51}$$

The first and second derivatives of $\beta(\mathbf{r})$ of eq. 49 with respect to internal coordinates are:

$$\frac{\partial \beta(\mathbf{r})}{\partial \mathbf{r}} = \sum_{k=1}^{N}\left(\frac{\partial w_k}{\partial \mathbf{r}} V'_{12}(\mathbf{r};k) + w_k \mathbf{g}_{12}(\mathbf{r};k)\right) \tag{52}$$

$$\frac{\partial^2\beta(\mathbf{r})}{\partial\mathbf{r}^2} = \sum_{k=1}^{N}\left(\frac{\partial^2 w_k}{\partial\mathbf{r}^2}V'_{12}(\mathbf{r};k) + \frac{\partial w_k}{\partial\mathbf{r}}\mathbf{g}_{12}(\mathbf{r};k)^{\mathsf{T}} + \mathbf{g}_{12}(\mathbf{r};k)\left(\frac{\partial w_k}{\partial\mathbf{r}}\right)^{\mathsf{T}} + w_k\mathbf{f}_{12}(\mathbf{r};k)\right), \tag{53}$$

where

$$\mathbf{g}_{12} \equiv \frac{\partial V'_{12}(\mathbf{r};k)}{\partial\mathbf{r}} = \frac{1}{2V'_{12}(\mathbf{r};k)}D^{(k)}\left(\mathbf{b}^{(k)} + \mathbf{C}^{(k)}\mathbf{\Delta r}^{(k)}\right)u(\mathbf{r};k)\left\{1 + nu(\mathbf{r};k)\left(\frac{\Delta}{T_{12}(\mathbf{r};k)}\right)^{2n}\right\} \tag{54}$$

and

$$\begin{aligned}\mathbf{f}_{12} &\equiv \frac{\partial^2 V'_{12}(\mathbf{r};k)}{\partial\mathbf{r}^2}\\ &= \frac{1}{V'_{12}(\mathbf{r};k)}\left[-\mathbf{g}_{12}(\mathbf{r};k)\mathbf{g}_{12}(\mathbf{r};k)^{\mathsf{T}} + \frac{1}{2}D^{(k)}\mathbf{C}^{(k)}u(\mathbf{r};k)\left\{nu(\mathbf{r};k)\left(\frac{\Delta}{T_{12}(\mathbf{r};k)}\right)^{2n} + 1\right\}\right.\\ &\quad\left. + \left\{\frac{D^{(k)}\left(\mathbf{b}^{(k)} + \mathbf{C}^{(k)}\mathbf{\Delta r}^{(k)}\right)nu(\mathbf{r};k)}{T_{12}(\mathbf{r};k)}\right\}^2\left(\frac{\Delta}{T_{12}(\mathbf{r};k)}\right)^{2n}\left\{nu(\mathbf{r};k)\left(\frac{\Delta}{T_{12}(\mathbf{r};k)}\right)^{2n} - (n-1)\right\}\right],\end{aligned} \tag{55}$$

where the coefficients $D^{(k)}$, $\mathbf{b}^{(k)}$, and $\mathbf{C}^{(k)}$ are given in eqs. 35-37. The derivatives of the weight function with respect to coordinates $\mathbf{r}$ required by eqs. 52 and 53 are the same as those given in eqs. 47 and 48. This approach is called Hermitian MCSI, and it is obsolete. The default values for Δ and n in the MCSI code are 1.6×10^{-4} E_h and 2, respectively.

The normalized weight function is:

$$w_k(\mathbf{s}) = \frac{\dfrac{1}{d_k(\mathbf{s})^4}}{\displaystyle\sum_{k=1}^{N+2}\frac{1}{d_k(\mathbf{s})^4}} \tag{56}$$

where d_k is the generalized distance between $\mathbf{s}$ and $\mathbf{s}^{(k)}$ defined as:

$$d_k(\mathbf{s}) = \sqrt{\sum_{\gamma=1}^{\Gamma}\left(s_\gamma - s_\gamma^{(k)}\right)^2}, \tag{57}$$

where $\mathbf{s} \equiv \{s_1, s_2, \ldots s_\gamma, \ldots s_\Gamma\}$ is a set of internal coordinates that is generally different from the set $\mathbf{r}$. Note that the sum in eq. 56 has two more points than the sum in eq. 40. The two extra points consist of one point in the region where V is assumed to be well approximated by V_{11} and another in the region where V is assumed to be well approximated by V_{22}. At both of these points V_{12} and V'_{12} are assumed to be zero, so these points do not occur in eq. 40. Thus eq. 40 actually corresponds to an $(N+2)$-point interpolation with N terms.

The first and second derivatives with respect to $\mathbf{s}$ of the weight function given in eq. 56 are currently obtained *numerically*. Since all operations except for this numerical intermediate step are analytic, the final MCSI derivatives may be called *semi-analytical*.

(vii) Find the eigenvalue V of eq. 39 and its derivatives in Cartesian coordinates. These are given by:

$$\begin{aligned}V(\mathbf{x}) =& \frac{1}{2}\left[V_{11}(\mathbf{x}) + V_{22}(\mathbf{x})\right.\\ &\left. - \left\{\left(V_{11}(\mathbf{x}) - V_{22}(\mathbf{x})\right)^2 + 4\beta^2(\mathbf{x})\right\}^{1/2}\right],\end{aligned} \tag{58}$$

where β^2 is equal (at all geometries where $\beta_o^2(\mathbf{r}) > -\Delta^2/4$) to β_o^2 obtained directly via the N-term Shepard interpolation, eq. 40, and

$$G_i = \frac{\partial V}{\partial x_i} = \frac{1}{2}\left[G_{11i} + G_{22i} - \frac{2\left(\dfrac{\partial\beta^2}{\partial x_i}\right) + (V_{11} - V_{22})(G_{11i} - G_{22i})}{\left\{(V_{11} - V_{22})^2 + 4\beta^2\right\}^{1/2}}\right] \tag{59}$$

$$F_{ij} = \frac{\partial^2 V}{\partial x_i \partial x_j}$$
$$= \frac{1}{2}\left[F_{11ij} + F_{22ij} - \frac{2\left(\frac{\partial^2 \beta^2}{\partial x_i \partial x_j}\right) + (V_{11}-V_{22})\left(F_{11ij}-F_{22ij}\right) + (G_{11i}-G_{22i})\left(G_{11j}-G_{22j}\right)}{\left\{(V_{11}-V_{22})^2 + 4\beta^2\right\}^{1/2}} \right.$$
$$\left. + \frac{\left\{2\left(\frac{\partial \beta^2}{\partial x_i}\right) + (V_{11}-V_{22})\left(G_{11i}-G_{22i}\right)\right\}\left\{2\left(\frac{\partial \beta^2}{\partial x_j}\right) + (V_{11}-V_{22})\left(G_{11j}-G_{22j}\right)\right\}}{\left\{(V_{11}-V_{22})^2 + 4\beta^2\right\}^{3/2}} \right]. \quad (60)$$

If matrix $\mathbf{V}$ is set to be Hermitian (old MCSI formalism), then β (ratehr than β^2) is calculated by eqs. 49 and 51, and the gradient and Hessian components of V with respect to Cartesian coordinates in terms of β are given by:

$$G_i = \frac{\partial V}{\partial x_i} = \frac{1}{2}\left[G_{11i} + G_{22i} - \frac{4\beta\left(\frac{\partial \beta}{\partial x_i}\right) + (V_{11}-V_{22})(G_{11i}-G_{22i})}{\left\{(V_{11}-V_{22})^2 + 4\beta^2\right\}^{1/2}} \right] \quad (61)$$

and

$$F_{ij} = \frac{\partial^2 V}{\partial x_i \partial x_j}$$
$$= \frac{1}{2}\left[F_{11ij} + F_{22ij} + \frac{\left\{4\beta\left(\frac{\partial \beta}{\partial x_i}\right) + (V_{11}-V_{22})\left(G_{11i}-G_{22i}\right)\right\}\left\{4\beta\left(\frac{\partial \beta}{\partial x_j}\right) + (V_{11}-V_{22})\left(G_{11j}-G_{22j}\right)\right\}}{\left\{(V_{11}-V_{22})^2 + 4\beta^2\right\}^{3/2}} \right.$$
$$\left. - \frac{4\left(\frac{\partial \beta}{\partial x_i}\right)\left(\frac{\partial \beta}{\partial x_j}\right) + 4\beta\left(\frac{\partial^2 \beta}{\partial x_i \partial x_j}\right) + (G_{11i}-G_{22i})\left(G_{11j}-G_{22j}\right) + (V_{11}-V_{22})\left(F_{11ij}-F_{22ij}\right)}{\left\{(V_{11}-V_{22})^2 + 4\beta^2\right\}^{1/2}} \right]. \quad (62)$$

IV. MCSI ALGORITHM (SYMMETRIZED)

The procedure for constructing a potential energy surface that is invariant with respect to the exchange of identical nuclei using MCSI is described in Ref. 4. The algorithm is summarized below:

(i) a) Select k = 1, 2, ..., N molecular geometries $\mathbf{x}^{(k)}$ to be used as electronic structure Shepard points, and b) calculate the energies $V^{(k)}$, gradients $\mathbf{G}^{(k)}$, and Hessians $\mathbf{F}^{(k)}$ at these geometries (e.g., using the *Gaussian* code).

(ii) For each of these data points $\mathbf{x}^{(k)}$ generate $m!$ symmetrically equivalent data sets $\{\mathbf{x}^{(k,i)}, \mathbf{G}^{(k,i)}, \mathbf{F}^{(k,i)}\}$, where

$$\mathbf{x}^{(k,i)} = \mathbf{P}^{(i)}\mathbf{x}^{(k)} \quad (63)$$

$$\mathbf{G}^{(k,i)} \equiv \frac{\partial}{\partial \mathbf{x}} V = \mathbf{P}^{(i)}\mathbf{G}^{(k)} \quad (64)$$

$$\mathbf{F}^{(k,i)} \equiv \frac{\partial^2}{\partial \mathbf{x}^2} V = \mathbf{P}^{(i)}\mathbf{F}^{(k)}\mathbf{P}^{(i)} \quad (65)$$

where $\mathbf{P}$ is the nuclear permutation operator that interchanges Cartesian coordinates of identical nuclei.

(iii) Define a set of $m!$ MM energies, gradients, and Hessians at point (k,i) by:

$$V^{(k,i)}_{\mathrm{MM},n} \equiv V_{nn}(\mathbf{x}^{(k,i)}) \quad (66)$$

$$\mathbf{G}^{(k,i)}_{\mathrm{MM},n} \equiv \frac{\partial}{\partial \mathbf{x}} V_{nn}\Big|_{\mathbf{x}=\mathbf{x}^{(k,i)}} \quad (67)$$

and

$$\mathbf{F}^{(k,i)}_{\mathrm{MM},n} \equiv \frac{\partial^2}{\partial \mathbf{x}^2} V_{nn}\Big|_{\mathbf{x}=\mathbf{x}^{(k,i)}} \quad (68)$$

for n = 1, 2; k = 1, 2, ..., N; i = 1, 2, ..., $m!$.

(iv) Define a symmetrized MM potential and its gradient and Hessian at point (k) (where a tilde denotes a symmetrization) by:

$$\tilde{V}^{(k)}_n = -\frac{1}{\alpha}\ln\left(\frac{1}{\sigma_{mm}}\sum_i^{m!} e^{-\alpha V^{(k,i)}_{\mathrm{MM},n}}\right), \quad (69)$$

where α is a parameter; σ_{mm}, which is called the symmetry

factor, is the number of times the lowest-energy MM configuration occurs among the $m!$ symmetrically equivalent MM configurations at a general geometry;

$$\tilde{\mathbf{G}}_n^{(k)} \equiv \frac{\partial}{\partial \mathbf{x}} \tilde{V_n}^{(k)} = \frac{\sum_i^{m!} \mathbf{G}_{\mathrm{MM},n}^{(k,i)} e^{-\alpha V_{\mathrm{MM},n}^{(k,i)}}}{\sum_i^{m!} e^{-\alpha V_{\mathrm{MM},n}^{(k,i)}}}, \tag{70}$$

and

$$\begin{aligned}\tilde{\mathbf{F}}_n^{(k)} \equiv & \frac{\partial^2}{\partial \mathbf{x}^2} \tilde{V_n}^{(k)} \\ = & \frac{\sum_i^{m!} \left(\mathbf{F}_{\mathrm{MM},n}^{(k,i)} - \alpha \mathbf{G}_{\mathrm{MM},n}^{(k,i)} \mathbf{G}_{\mathrm{MM},n}^{(k,i)\mathsf{T}} \right) e^{-\alpha V_{\mathrm{MM},n}^{(k,i)}}}{\sum_i^{m!} e^{-\alpha V_{\mathrm{MM},n}^{(k,i)}}} \\ & + \alpha \tilde{\mathbf{G}}_n^{(k)} \tilde{\mathbf{G}}_n^{(k)\mathsf{T}}. \end{aligned} \tag{71}$$

Notice that the symmetrized MM potential is dominated by the σ_{mm} lowest-energy MM configurations among the $m!$ permutations of the labels on the identical atoms. The parameter α controls the rate of switching between different dominant configurations in regions that separate the low-energy regions corresponding to the differently permuted coordinates.

(v) Generate $m!$ values of $\tilde{\mathbf{G}}_n^{(k,i)}$ and $\tilde{\mathbf{F}}_n^{(k,i)}$ from each $\tilde{\mathbf{G}}^{(k)}$, and $\tilde{\mathbf{F}}^{(k)}$ by applying $\mathbf{P}^{(i)}$, as in step (ii).

(vi) Transform $\mathbf{G}^{(k,i)}$, $\mathbf{F}^{(k,i)}$, $\tilde{\mathbf{G}}_n^{(k,i)}$, and $\tilde{\mathbf{F}}_n^{(k,i)}$ to the set of internal coordinates $\mathbf{r}$ by the Wilson B matrix and C tensor. This yields:

$$\mathbf{g}^{(k,i)} \equiv \frac{\partial}{\partial \mathbf{r}} V \Big|_{\mathbf{r}=\mathbf{r}(\mathbf{x}^{(k,i)})} \tag{72}$$

$$\mathbf{f}^{(k,i)} \equiv \frac{\partial^2}{\partial \mathbf{r}^2} V \Big|_{\mathbf{r}=\mathbf{r}(\mathbf{x}^{(k,i)})} \tag{73}$$

$$\tilde{\mathbf{g}}_n^{(k,i)} \equiv \frac{\partial}{\partial \mathbf{r}} \tilde{V}_n \Big|_{\mathbf{r}=\mathbf{r}(\mathbf{x}^{(k,i)})} \tag{74}$$

$$\tilde{\mathbf{f}}_n^{(k,i)} \equiv \frac{\partial^2}{\partial \mathbf{r}^2} \tilde{V}_n \Big|_{\mathbf{r}=\mathbf{r}(\mathbf{x}^{(k,i)})} \tag{75}$$

(vii) Define a matrix $\mathbf{V}^{(k,i)}$ at each geometry (k,i) by

$$\mathbf{V}^{(k,i)}(\mathbf{r}) = \begin{pmatrix} \tilde{V_1}^{(k,i)}(\mathbf{r}) & V_{12}(\mathbf{r};k,i) \\ V_{12}(\mathbf{r};k,i) & \tilde{V_2}^{(k,i)}(\mathbf{r}) \end{pmatrix} \tag{76}$$

and construct Taylor series expansions of V_{12} around each data point (k,i) using Taylor series reversion in the same way as in eqs. 32 and 33 for nonsymmetrized calculations, with the only difference being that (k) is replaced by (k,i).

(viii) Calculate the Taylor series coefficients D, $\mathbf{b}$, and $\mathbf{C}$ (as in eqs. 35-37) for each symmetrically equivalent data point (k,i):

$$D^{(k,i)} = \left(\tilde{V}_1^{(k,i)} - V^{(k,i)} \right) \left(\tilde{V}_2^{(k,i)} - V^{(k,i)} \right) \tag{77}$$

$$\mathbf{b}^{(k,i)} = \frac{\tilde{\mathbf{g}}_1^{(k,i)} - \mathbf{g}^{(k,i)}}{\tilde{V}_1^{(k,i)} - V^{(k,i)}} - \frac{\tilde{\mathbf{g}}_2^{(k,i)} - \mathbf{g}^{(k,i)}}{\tilde{V}_2^{(k,i)} - V^{(k,i)}} \tag{78}$$

$$\begin{aligned}\mathbf{C}^{(k,i)} = & \left(1/D^{(k,i)} \right) \Big[\left(\tilde{\mathbf{g}}_1^{(k,i)} - \mathbf{g}^{(k,i)} \right) \left(\tilde{\mathbf{g}}_2^{(k,i)} - \mathbf{g}^{(k,i)} \right) \\ & + \left(\tilde{\mathbf{g}}_2^{(k,i)} - \mathbf{g}^{(k,i)} \right) \left(\tilde{\mathbf{g}}_1^{(k,i)} - \mathbf{g}^{(k,i)} \right)^{\mathsf{T}} \Big] \\ & + \frac{\tilde{\mathbf{f}}_1^{(k,i)} - \mathbf{f}^{(k,i)}}{\tilde{V}_1^{(k,i)} - V^{(k,i)}} + \frac{\tilde{\mathbf{f}}_2^{(k,i)} - \mathbf{f}^{(k,i)}}{\tilde{V}_2^{(k,i)} - V^{(k,i)}} \end{aligned} \tag{79}$$

The second-order Taylor series of $(V_{12})^2$, $(T_{12})^2$, for each (k,i) at an arbitrary geometry $\mathbf{r} = \mathbf{r}(\mathbf{x})$ can now be written as:

$$\begin{aligned}[T_{12}]^2 \equiv & [V_{12}(\mathbf{r};k,i)]^2 \\ = & D^{(k,i)} \Bigg(1 + \mathbf{b}^{(k,i)\mathsf{T}} \Delta \mathbf{r}^{(k,i)} \\ & + \frac{1}{2} \Delta \mathbf{r}^{(k,i)\mathsf{T}} \mathbf{C}^{(k,i)} \Delta \mathbf{r}^{(k,i)} \Bigg) \end{aligned} \tag{80}$$

(ix) Define the matrix $\mathbf{V}$ at the input geometry $\mathbf{x}$ by

$$\mathbf{V}(\mathbf{x}) = \begin{pmatrix} \tilde{V}_1(\mathbf{x}) & \beta(\mathbf{x}) \\ \beta(\mathbf{x}) & \tilde{V}_2(\mathbf{x}) \end{pmatrix}. \tag{81}$$

The lowest-energy eigenvalue of this matrix is the MCSI potential energy function. The diagonal matrix elements $\tilde{V}_n(\mathbf{x})$ and their derivatives, $\tilde{\mathbf{G}}_n(\mathbf{x})$ and $\tilde{\mathbf{F}}_n(\mathbf{x})$, are obtained as follows: First we define

$$V_{\mathrm{MM},n}^{(j)} \equiv V_{nn}^{(j)}(\mathbf{x}) \qquad j = 1, ..., m! \tag{82}$$

$$\mathbf{G}_{\mathrm{MM},n}^{(j)} \equiv \frac{\partial}{\partial \mathbf{x}} V_{nn}^{(j)} \qquad j = 1, ..., m! \tag{83}$$

and

$$\mathbf{F}_{\mathrm{MM},n}^{(j)} \equiv \frac{\partial^2}{\partial \mathbf{x}^2} V_{nn}^{(j)} \qquad j = 1, ..., m! \tag{84}$$

where each value of j corresponds to one of the $m!$ connectivity patterns. Then, $\tilde{V}_n(\mathbf{x})$, $\tilde{\mathbf{G}}_n(\mathbf{x})$, and $\tilde{\mathbf{F}}_n(\mathbf{x})$ are calculated as:

$$\tilde{V}_n(\mathbf{x}) = -\frac{1}{\alpha} \ln \left(\frac{1}{\sigma_{mm}} \sum_j^{m!} e^{-\alpha V_{\mathrm{MM},n}^{(j)}(\mathbf{x})} \right), \tag{85}$$

$$\tilde{\mathbf{G}}_n(\mathbf{x}) = \frac{\sum_j^{m!} \mathbf{G}^{(j)}_{\mathrm{MM},n}(\mathbf{x}) e^{-\alpha V^{(j)}_{\mathrm{MM},n}(\mathbf{x})}}{\sum_j^{m!} e^{-\alpha V^{(j)}_{\mathrm{MM},n}(\mathbf{x})}}, \tag{86}$$

and

$$\widetilde{\mathbf{F}}_n(\mathbf{x}) = \frac{\sum_j^{m!} \left(\mathbf{F}^{(j)}_{\mathrm{MM},n}(\mathbf{x}) - \alpha \mathbf{G}^{(j)}_{\mathrm{MM},n}(\mathbf{x}) \mathbf{G}^{(j)}_{\mathrm{MM},n}{}^{\mathsf{T}}(\mathbf{x}) \right) e^{-\alpha V^{(j)}{}_{\mathrm{MM},n}(\mathbf{x})}}{\sum_j^{m!} e^{-\alpha V^{(j)}{}_{\mathrm{MM},n}(\mathbf{x})}} + \alpha \widetilde{\mathbf{G}}_n \widetilde{\mathbf{G}}_n^{\mathsf{T}}, \tag{87}$$

where $V^{(j)}_{\mathrm{MM},n}(\mathbf{x})$, $\mathbf{G}^{(j)}_{\mathrm{MM},n}(\mathbf{x})$, and $\mathbf{F}^{(j)}_{\mathrm{MM},n}(\mathbf{x})$ are sets of $m!$ MM energies, gradients, and Hessians at the geometry $\mathbf{x}$.

As in the algorithm for constructing nonsymmetrized PESs, two approaches are available to obtain the off-diagonal elements β. In the first case, one carries out the Shepard interpolation of T^2_{12} of eq. 80 directly:

$$\beta^2_o(\mathbf{r}) = \sum_{k=1}^{N} \sum_{i=1}^{m!} w_{ki}(\mathbf{r}) T^2_{12}(\mathbf{r}; k, i), \tag{88}$$

where w_{ki} are normalized weights. The value of β^2 at a geometry $\mathbf{r}$ is then approximated in the same way as in eqs. 41 and 42 in the nonsymmetrized algorithm.

The first and second derivatives of β^2 with respect to the coordinates $\mathbf{r}$ are the same as the derivatives of β_o for all $\beta^2_o \geq -\Delta^2/4$. These derivatives are given by

$$\mathbf{g}(\mathbf{r}) \equiv \frac{\partial \beta^2(\mathbf{r})}{\partial \mathbf{r}} = \sum_{k=1}^{N} \sum_{i=1}^{m!} \left[\frac{\partial w_{ki}}{\partial \mathbf{r}} T^2_{12}(\mathbf{r}; k, i) + w_{ki} \mathbf{g}_{12}(\mathbf{r}; k, i) \right] \tag{89}$$

and

$$\mathbf{f}(\mathbf{r}) \equiv \frac{\partial^2 \beta^2(\mathbf{r})}{\partial \mathbf{r}^2} = \sum_{k=1}^{N} \sum_{i=1}^{m!} \left(\frac{\partial^2 w_{ki}}{\partial \mathbf{r}^2} T^2_{12}(\mathbf{r}; k; i) + \frac{\partial w_{ki}}{\partial \mathbf{r}} \mathbf{g}_{12}(\mathbf{r}; k, i)^{\mathsf{T}} + \mathbf{g}_{12}(\mathbf{r}; k, i) \left(\frac{\partial w_{ki}}{\partial \mathbf{r}} \right)^{\mathsf{T}} + w_{ki} \mathbf{f}_{12}(\mathbf{r}; k, i) \right), \tag{90}$$

where

$$\begin{aligned} \mathbf{g}_{12}(\mathbf{r}; k, i) \equiv & \frac{\partial T^2_{12}(\mathbf{r}; k, i)}{\partial \mathbf{r}} \\ = & D^{(k,i)} \left(\mathbf{b}^{(k,i)} + \mathbf{C}^{(k,i)} \boldsymbol{\Delta} \mathbf{r} \right) \end{aligned} \tag{91}$$

and

$$\mathbf{f}_{12}(\mathbf{r}; k, i) \equiv \frac{\partial^2 T^2_{12}(\mathbf{r}; k, i)}{\partial \mathbf{r}^2} = \mathbf{C}^{(k,i)} \tag{92}$$

The first and second derivatives of the weight function with respect to coordinates $\mathbf{r}$ that are required by eqs. 89 and 90 are given by:

$$\frac{\partial w_k}{\partial r_\alpha} = \sum_{\gamma=1}^{\Gamma} \frac{\partial w_k}{\partial s_\gamma} \frac{\partial s_\gamma}{\partial r_\alpha} \tag{93}$$

and

$$\frac{\partial^2 w_k}{\partial r_\alpha \partial r_\beta} = \sum_{\gamma=1}^{\Gamma} \sum_{\gamma=1'}^{\Gamma} \frac{\partial s_\gamma}{\partial r_\alpha} \frac{\partial^2 w_k}{\partial s_\gamma \partial s_{\gamma'}} \frac{\partial s_{\gamma'}}{\partial r_\beta} + \sum_{\gamma=1}^{\Gamma} \frac{\partial w_k}{\partial s_\gamma} \frac{\partial^2 s_\gamma}{\partial r_\alpha \partial r_\beta}, \tag{94}$$

where $\mathbf{r}$ and $\mathbf{s}$ are the sets of the internal coordinates used in Shepard interpolation and in calculations of the weight function.

In the other approach, called Hermitian MCSI, the value of β and its first and second derivatives with respect to the internal coordinates are obtained as follows:

The Shepard interpolation step yields:

$$\beta(\mathbf{r}) = \sum_{k=1}^{N} \sum_{i=1}^{m!} w_{ki}(\mathbf{r}) V'_{12}(\mathbf{r}; k, i), \tag{95}$$

where w_{ki} are normalized weights, and V'_{12} is defined by

$$V'_{12}(\mathbf{r}; k, i) = \sqrt{T^2_{12}(\mathbf{r}; k, i) u(\mathbf{r}; k, i)}, \tag{96}$$

where $T_{12}(\mathbf{r},k,i)^2$ is given in eq. 80, and

$$u(\mathbf{r},k,i) = \begin{cases} \dfrac{1}{1+(\Delta/T_{12}(\mathbf{r};k,i))^{2n}}; & T_{12}(\mathbf{r};k,i)^2 > 0 \\ 0; & \text{otherwise.} \end{cases} \tag{97}$$

The gradient and Hessian of β of eq. 81 with respect to internal coordinates are given by:

$$\mathbf{g}(\mathbf{r}) \equiv \frac{\partial\beta(\mathbf{r})}{\partial\mathbf{r}} = \sum_{k=1}^{N}\sum_{i=1}^{m!}\left[\frac{\partial w_{ki}}{\partial\mathbf{r}}V'_{12}(\mathbf{r};k,i) + w_{ki}\mathbf{g}_{12}(\mathbf{r};k,i)\right] \tag{98}$$

$$\mathbf{f}(\mathbf{r}) \equiv \frac{\partial^2\beta(\mathbf{r})}{\partial\mathbf{r}^2} = \sum_{k=1}^{N}\sum_{i=1}^{m!}\left(\frac{\partial^2 w_{ki}}{\partial\mathbf{r}^2}V'_{12}(\mathbf{r};k,i) + \frac{\partial w_{ki}}{\partial\mathbf{r}}\mathbf{g}_{12}(\mathbf{r};k,i)^{\mathsf{T}} + \mathbf{g}_{12}(\mathbf{r};k,i)\left(\frac{\partial w_{ki}}{\partial\mathbf{r}}\right)^{\mathsf{T}} + w_{ki}\mathbf{f}_{12}(\mathbf{r};k,i)\right), \tag{99}$$

where

$$\mathbf{g}_{12}(\mathbf{r};k,i) \equiv \frac{\partial V'_{12}(\mathbf{r};k,i)}{\partial\mathbf{r}} = \frac{1}{2V'_{12}(\mathbf{r};k,i)}D^{(k,i)}\left(\mathbf{b}^{(k,i)} + \mathbf{C}^{(k,i)}\boldsymbol{\Delta}\mathbf{r}^{(k,i)}\right)u(\mathbf{r};k,i)\left(1+\left(\frac{\Delta}{T_{12}(\mathbf{r};k,i)}\right)^{2n}nu(\mathbf{r};k,i)\right) \tag{100}$$

and

$$\begin{aligned}\mathbf{f}_{12} &\equiv \frac{\partial^2 V'_{12}(\mathbf{r};k,i)}{\partial\mathbf{r}^2} \\ &= \frac{1}{V'_{12}(\mathbf{r};k,i)}\left[-\mathbf{g}_{12}(\mathbf{r};k,i)\mathbf{g}_{12}(\mathbf{r};k,i)^{\mathsf{T}} + \frac{1}{2}D^{(k,i)}\mathbf{C}^{(k,i)}u(\mathbf{r};k,i)\left\{nu(\mathbf{r};k,i)\left(\frac{\Delta}{T_{12}(\mathbf{r};k,i)}\right)^{2n}+1\right\}\right. \\ &\quad \left. + \left\{\frac{D^{(k,i)}\left(\mathbf{b}^{(k,i)}+\mathbf{C}^{(k,i)}\boldsymbol{\Delta}\mathbf{r}^{(k,i)}\right)nu(\mathbf{r};k,i)}{T_{12}(\mathbf{r};k,i)}\right\}^2\left(\frac{\Delta}{T_{12}(\mathbf{r};k,i)}\right)^{2n}\left\{nu(\mathbf{r};k,i)\left(\frac{\Delta}{T_{12}(\mathbf{r};k,i)}\right)^{2n}-(n-1)\right\}\right]\end{aligned} \tag{101}$$

As in the nonsymmetrized MCSI calculations, the first and second derivatives with respect to $\mathbf{r}$ of the weight function are obtained numerically. Since all operations except for this numerical intermediate step are analytic, the final MCSI derivatives may be called semi-analytical.

These derivatives of β are then transformed from the internal coordinates $\mathbf{r}$ to Cartesian coordinates by using the transformation matrices saved in an earlier step, in the same fashion as in the formalism for nonsymmetrized potential energy surfaces.

The normalized weight function of eq. 95 is:

$$w_{ki}(\mathbf{s}) = \frac{\dfrac{Y_{ki}(\mathbf{r})}{d_{ki}^4(\mathbf{s})}}{\displaystyle\sum_{k=1}^{(N+2)}\sum_{i=1}^{m!}\frac{Y_{ki}(\mathbf{r})}{d_{ki}^4(\mathbf{s})}} \tag{102}$$

where d_{ki} is the generalized distance between $\mathbf{s}$ and $\mathbf{s}^{(k,i)}$ defined as:

$$d_{ki}(\mathbf{s}) = \sqrt{\sum_{\gamma=1}^{\Gamma m!}\left(s_\gamma - s_\gamma^{(k,i)}\right)^2}, \tag{103}$$

where $\mathbf{s} \equiv \{s_1, s_2, \ldots s_\gamma, \ldots s_\Gamma\}$. The current implementation only supports the following cases: (i) set $\mathbf{s}$ is the same as set $\mathbf{r}$, and (ii) set $\mathbf{s}$ is a subset of $\mathbf{r}$. The scaling coefficients $Y_{ki}(\mathbf{r})$ are chosen such that the reactive system is described by pure molecular mechanics in the asymptotic regions. In particular, we define Y_{ki} at a geometry $\mathbf{r}$ as:

$$Y_{ki}(\mathbf{r}) = \frac{1}{1+\left(\dfrac{T_{12}^2(\mathbf{r};k,i) - D^{(k,i)}}{A^2}\right)^{\mu}}, \tag{104}$$

where A and μ are parameters; the default values are $A = \sqrt{6}\times 10^{-4}\ E_h$, $\mu = 4$.

(x) Find the eigenvalue $\mathbf{V}$ of eq. 81 and its derivatives in Cartesian coordinates. The lowest eigenvalue of eq 81 is given by

$$\begin{aligned}V =& \frac{1}{2}\left[\widetilde{V}_1(\mathbf{q}(\mathbf{x})) + \widetilde{V}_2(\mathbf{q}(\mathbf{x}))\right. \\ &\left. - \left\{\left(\widetilde{V}_1(\mathbf{q}(\mathbf{x})) - \widetilde{V}_2(\mathbf{q}(\mathbf{x}))\right)^2 + 4\beta^2(\mathbf{r}(\mathbf{x}))\right\}^{1/2}\right]\end{aligned} \tag{105}$$

where $\tilde{V}_n$ are the symmetrized uninterpolated MM potentials given by eq. 69, and β^2 is equal (at all geometries where $\beta_o^2(\mathbf{r}) > -\Delta^2/4$) to β_o^2 obtained via the $m!N$-term Shepard interpolation, eq. 88. The gradient and Hessian components of V with respect to Cartesian coordinates are given by:

$$G_i = \frac{\partial V}{\partial x_i} = \frac{1}{2}\left[\widetilde{G}_{1i}(\mathbf{x}) + \widetilde{G}_{2i}(\mathbf{x}) - \frac{2\left(\frac{\partial \beta^2(\mathbf{r}(\mathbf{x}))}{\partial x_i}\right) + \left(\widetilde{V}_1(\mathbf{x}) - \widetilde{V}_2(\mathbf{x})\right)\left(\widetilde{G}_{1i}(\mathbf{x}) - \widetilde{G}_{2i}(\mathbf{x})\right)}{\left\{\left(\widetilde{V}_1(\mathbf{x}) - \widetilde{V}_2(\mathbf{x})\right)^2 + 4\beta^2(\mathbf{r}(\mathbf{x}))\right\}^{1/2}}\right] \tag{106}$$

$$\begin{aligned} F_{ij} &= \frac{\partial^2 V}{\partial x_i \partial x_j} \\ &= \frac{1}{2}\left[\widetilde{F}_{1ij}(\mathbf{x}) + \widetilde{F}_{2ij}(\mathbf{x}) - \frac{2\left(\frac{\partial^2 \beta^2}{\partial x_i \partial x_j}\right) + \left(\widetilde{V}_1 - \widetilde{V}_2\right)\left(\widetilde{F}_{1ij} - \widetilde{F}_{2ij}\right) + \left(\widetilde{G}_{1i} - \widetilde{G}_{2i}\right)\left(\widetilde{G}_{1j} - \widetilde{G}_{2j}\right)}{\left\{\left(\widetilde{V}_1 - \widetilde{V}_2\right)^2 + 4\beta^2\right\}^{1/2}}\right. \\ &\quad \left. + \frac{\left\{2\left(\frac{\partial \beta^2}{\partial x_i}\right) + \left(\widetilde{V}_1 - \widetilde{V}_2\right)\left(\widetilde{G}_{1i} - \widetilde{G}_{2i}\right)\right\}\left\{2\left(\frac{\partial \beta^2}{\partial x_j}\right) + \left(\widetilde{V}_1 - \widetilde{V}_2\right)\left(\widetilde{G}_{1j} - \widetilde{G}_{2j}\right)\right\}}{\left\{\left(\widetilde{V}_1 - \widetilde{V}_2\right)^2 + 4\beta^2(\mathbf{r}\right\}^{3/2}}\right]. \end{aligned} \tag{107}$$

If $\mathbf{H}$ is Hermitian, then the components of the gradient vector and the Hessian matrix of V are calculated (in terms of β) as

$$G_i = \frac{\partial V}{\partial x_i} = \frac{1}{2}\left[\widetilde{G}_{1i}(\mathbf{x}) + \widetilde{G}_{2i}(\mathbf{x}) - \frac{4\left(\frac{\partial \beta}{\partial x_i}\right) + \left(\widetilde{V}_1(\mathbf{x}) - \widetilde{V}_2(\mathbf{x})\right)\left(\widetilde{G}_{1i}(\mathbf{x}) - \widetilde{G}_{2i}(\mathbf{x})\right)}{\left\{\left(\widetilde{V}_1(\mathbf{x}) - \widetilde{V}_2(\mathbf{x})\right)^2 + 4\beta^2\right\}^{1/2}}\right] \tag{108}$$

and

$$\begin{aligned} F_{ij} &= \frac{\partial^2 V}{\partial x_i \partial x_j} \\ &= \frac{1}{2}\left[\widetilde{F}_{1ij} + \widetilde{F}_{2ij} + \frac{\left\{4\beta\left(\frac{\partial \beta}{\partial x_i}\right) + \left(\widetilde{V}_1 - \widetilde{V}_2\right)\left(\widetilde{G}_{1i} - \widetilde{G}_{2i}\right)\right\}\left\{4\beta\left(\frac{\partial \beta}{\partial x_j}\right) + \left(\widetilde{V}_1 - \widetilde{V}_2\right)\left(\widetilde{G}_{1j} - \widetilde{G}_{2j}\right)\right\}}{\left\{\left(\widetilde{V}_1 - \widetilde{V}_2\right)^2 + 4\beta^2\right\}^{3/2}}\right. \\ &\quad \left. - \frac{4\left(\frac{\partial \beta}{\partial x_i}\right)\left(\frac{\partial \beta}{\partial x_j}\right) + 4\beta\left(\frac{\partial^2 \beta}{\partial x_i \partial x_j}\right) + \left(\widetilde{G}_{1i} - \widetilde{G}_{2i}\right)\left(\widetilde{G}_{1j} - \widetilde{G}_{2j}\right) + \left(\widetilde{V}_1 - \widetilde{V}_2\right)\left(\widetilde{F}_{1ij} - \widetilde{F}_{2ij}\right)}{\left\{\left(\widetilde{V}_1 - \widetilde{V}_2\right)^2 + 4\beta^2\right\}^{1/2}}\right]. \end{aligned} \tag{109}$$

where β is obtained via the $m!N$-term Shepard interpolation of modified T_{12}^2, eqs. 95-97.

V. NOTES ON HOW TO TREAT SHALLOW POTENTIAL ENERGY SURFACES

The MCSI and EE-MCSI methods described above can in principle reproduce any type of reactive potential energy surface (PES) with good accuracy. One can raise the accuracy by adding Shepard points with an appropriate weight function at any locations where the MCSI potential energy function does not reproduce the reference PES well. Generally, the convergence is very fast due to the presence of the molecular mechanics force field in the diagonal elements. Tyically no more than 10 Shepard points are required to describe the PES with an error of less than 1.0 kcal/mol. However, when the system has a shallow PES in some directions, e.g. along dihedral angles, the convergence might be very slow because of the quadratic expansion of

$[T_{12}(\mathbf{r};k)]^2$ (eq. 38). For such directions, the diagonal element changes gradually, while $[T_{12}]^2$ can rapidly change as $\mathbf{\Delta r}^{(k)}$ becomes large. As a result, the MCSI PES might have large deviations from the accurate one along such directions. This problem does not appear if $\mathbf{\Delta r}^{(k)}$ is always small in the calculations such as when one concentrates only on regions near a minimum energy structure or near a minimum energy path. However, in cases such as molecular dynamics simulations, the problem can become severe because the calculations sample a wide region.

To overcome this problem, two options are available in the current version of MCSI (Ref 36). First, eq. 38 is modified for dihedral angles as follows,

$$[T_{12}(\mathbf{r};k)]^2 = D^{(k)}\Big\{1 + \mathbf{b}^{(k)\mathsf{T}}\mathbf{S}(\Delta\mathbf{r}^{(k)}) + \frac{1}{2}\mathbf{S}(\Delta\mathbf{r}^{(k)})^{\mathsf{T}}\mathbf{C}^{(k)}\mathbf{S}(\Delta\mathbf{r}^{(k)})\Big\}, \quad (110)$$

where

$$\mathbf{S}(\Delta\mathbf{r}^{(k)})^{\mathsf{T}} = \left(S(\Delta r_1^{(k)})\ \ S(\Delta r_2^{(k)})\ \ \cdots\ \ S(\Delta r_\Gamma^{(k)})\right), \quad (111)$$

and

$$S(\Delta r_\alpha^{(k)}) = \begin{cases} \dfrac{\sin\left(m_\alpha \Delta r_\alpha^{(k)}\right)}{m_\alpha}; & \text{if } r_\alpha \text{ is dihedral angle} \\ \Delta r_\alpha^{(k)}; & \text{otherwise.} \end{cases} \quad (112)$$

where m_α is the local periodicity of dihedral angle α. We recommend that m_α be determined physically; e.g, $m_\alpha = 3$ for dihedral angles describing internal rotation around a C-C single bond, and $m_\alpha = 2$ for those describing internal rotation around a C-C double bond. Note that the coefficients of the Taylor expansion, $D^{(k)}$, $\mathbf{b}^{(k)}$, and $\mathbf{C}^{(k)}$ are the same as in the original equation because the first and second derivatives of $\mathbf{S}(\Delta\mathbf{r}^{(k)})$ at $\Delta\mathbf{r}^{(k)} = \mathbf{0}$ are unchanged. This modification greatly reduces the error of the PES along dihedral angles. Note also that eq. 110 itself satisfies periodic boundary conditions with respect to dihedral angles, while the original equation does not, although the PES generated by the latter can satisfy periodic boundary condition by adding Shepard points along dihedral angles. This option can be specified by the ICSHEPARD keyword in the `esp.fu85` input file. The function S can also be used in the calculating the generalized distance $d_k(\mathbf{s})$ (eq. 57).

The second option is to simply set the quadratic coefficients of the Taylor expansion involving such directions equal to zero, namely $C_{\alpha\beta}^{(k)} = 0$ ($\beta = 1, 2, \cdots, \Gamma$) if the PES along internal coodinates α is problematic. This option can be used for any type of internal coordinates. This procedure is effective when the molecular mechanics force field in the diagonal elements describes the reference PES well, or the PES along such directions are not important for the reaction. Note that if all the quadratic coefficients are set equal to zero, it becomes gradient-based MCSI (Ref. 8). This option can be specified by the ICSHZERO keyword in the `esp.fu83` input file. In the current version of MCSI, one can also set the linear and quadratic coefficients of the Taylor expansion for an internal coordinate equal to zero, which can be specified by the ICSAZERO keyword in the `esp.fu83` input file.

VI. DISTRIBUTION

MCSI is a combination of two packages of code:

- TINKER 3.5mn5, which is a locally modified version of the TINKER program. (The modifications are listed in Sec. XIII A of this manual.) Because TINKER was re-organized by its author (J.W.P.) in version 3.6, it is not straightforward to incorporate the most recent version of TINKER into MCSI. Therefore we include (with permission from J. W. Ponder) a locally modified version of TINKER–version 3.5 in the MCSI distribution. The current modification is called TINKER–version 3.5mn5.
- MCT module, which is a set of subroutines that control and carry out SCMM and MCSI calculations by interfacing with TINKER 3.5mn3. The mct module also contains a number of modified TINKER subroutines that are used in MCSI instead of the original TINKER subroutines.

The use of the MCSI program requires a specific license. Further information can be obtained at:

```
http://comp.chem.umn.edu/mcsi
```

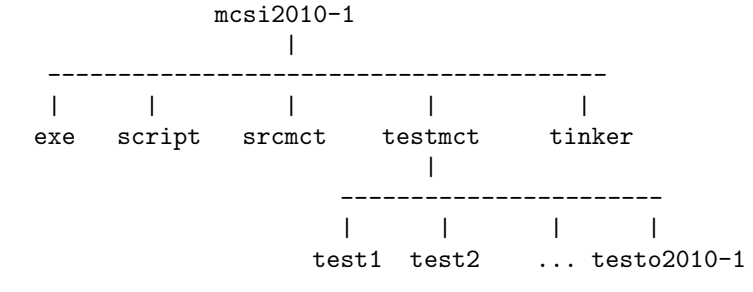
The program is distributed as a compressed tar file named `mcsi2010-1.tar.gz`. To uncompress, enter:

```
gunzip mcsi2010-1.tar.gz
```

The uncompressed file will be named `mcsi2010-1.tar`. After extracting the files from the tar file, which can be done with the command:

```
tar -xvf mcsi2010-1.tar
```

a new directory, `mcsi2010-1`, is created. This directory contains all the files included in the distribution package, which are located in subdirectories according to the following tree structure:



- `exe`: no files. (After the compilation, the executable will be moved to this directory.)

- `script`: 6 files containing compilation scripts for mcsi:
  `compile.gfortran, compile.gfortran.sym, compile.ifort, compile.ifort.sym, compile.pgf77, compile.pgf77.sym`.

- `srcmct`: 28 files comprising the mct module: `eemcmm.inc, emsrc1.f, energ.f, field.f, getkey.f, getprm.f, grads.f, hesss.f, initial.f, main.f, mccomm.inc, mcfset.f, mcparm.i, mcparm.inc, mcparm.inc.sym mcperc.inc, mcrsmldum.f, mcsrc1.f, mcsrc2.f, mcsrc3.f, mcsrc4.f, mcsrc5.f, mcsrc6.f, mcsrc7.f, mcsrc8.f, piscf.f, readfile.f, sizes.i`.

- `testmct`: 4 files: `run_all, run_t_all, getnew`, and `compnew`, and 25 subdirectories: 25 directories (`test1, test2`, ..., and `test25`) with the input files for the test calculations, one directory (`testo2010-1`) with the distributed outputs of the test runs, and one directory (`testonew`) without files. Running script `getnew` will collect the most recent outputs for all test calculations into directory `testonew`, and subsequently running script `compnew` will generate files for comparisons between these newly obtained and those distributed outputs for all test runs.

- `tinker`: The directory `tinker` has the following structure:

```
                    tinker
                      |
 ---------------------------------------
  |    |     |       |      |     |      |
bin  doc  ibmr6k  params  sgi  source  test
```

- `bin`: no files. (The binary executable files for TINKER 3.5mn5 will be located here.)

- `doc`: 9 files: `README, guide.txt, license.txt, spine.ps, summary.txt, guide.ps.Z, license.ps, logo.ps.Z, summary.ps`. These files are the original documentation files from TINKER 3.5.

- `ibmr6k`: 8 files: `README, clock.f, library.make, listing.make, calendar.f, compile.make, link.make, rename.make`. These files can be used for compiling TINKER–version 3.5mn3 as a stand-alone program on IBM RS/6000 machines.

- `sgi`: This directory contains the following subdirectories:

  – `irix5.3`: 10 files: `README, debug.make, link.make, listing.make, compile.make, library.make, link.make, rename.make, clock.f, calendar.f`

  – `irix6.2`: 8 files: `README, compile.make, library.make, link.make, listing.make, rename.make, clock.f, calendar.f`

  The files in these subdirectories can be used for compiling TINKER–version 3.5mn5 as a stand-alone program on SGI workstations under IRIX 5.3 or IRIX 6.2, respectively.

- `params`: 16 files: `README, merck.prm, mm3pro.prm, smooth.prm, amber.prm, emr.prm, mm2.prm, mmffpro.prm, tinker.prm, blank.prm, hoch.prm, charmm27t35.prm, mm3.prm, mmp_old.prm, opls.prm, water.prm`. These files are the parameter files for the supported force fields. We note that only the MM3 force field has been tested with MCSI.

- `source`: The source directory contains all the subroutines and include files for the standalone tinker 3.5mn5 program.

- `test`: Set of input files for testing TINKER 3.5mn5.

VII. INSTALLATION

First the user should obtain a licensed copy of the MCSI package. After downloading, uncompressing, and untaring the file, the MCSI package should appear as a directory, as described in Sec. VI. The MCSI package also includes a locally modified version of the TINKER program.

There are a few variables that have to be set large enough to accommodate the system or systems to be studied. Two of these variables are `maxatm` in the `mcparm.i` include file and `mcmesp` in the `mcparm.inc` include file in the srcmct directory of MCSI. The maxatm variable indicates the maximum number of atoms allowed in the executable, and the `mcmesp` variable indicates the maximum number of electronic structure theory Shepard points allowed in the executable. The first of these variables is set to a value of 40 and the second to a value of 20 in the distributed version of the code. The remaining variables in the include files can be changed according to the characteristics of the system and computer resources, but usually the default values are good enough.

The installation of the MCSI program can be carried out by running a C shell script, specifically, `compile.gfortran` for GNU Fortran compiler, `compile.ifort` for Intel Fortran compiler, `compile.pgf77` for PGI Fortran compiler. To run the script, the user should also give the path of the `mcsi2010-1` directory. For example, if the MCSI2010-1 path is `/usr/mcsi2010-1` then the command should be

```
compile.gfortran /usr/mcsi2010-1
```

for GNU Fortran compiler,

```
compile.ifort /usr/mcsi2010-1
```

for Intel Fortran compiler, or

```
compile.pgf77 /usr/mcsi2010-1
```

for PGI Fortran compiler. If one needs symmetrized MCSI calculations in addition to nonsymmetrized ones, use compile.XXX.sym instead of compile.XXX (XXX is the compiler name). Note that symmetrized MCSI calculations require a large amount of memory. These scripts will compile the source code and will generate an executable file. In Sec. XII we list the computers and operating systems on which the code has been tested. The installation of MCSI with a different kind of compiler or operating system should be straightforward, requiring only changes in the compiler and loader options in any one of the scripts, `compile.gfortran, compile.ifort`, or `compile.pgf77`. After installation, the executable file is put into the `exe` subdirectory of the `mcsi2010-1` directory.

The user can also compile a set of files from the TINKER 3.5mn5 suite as a stand-alone program. These files are described in the tinker manual in the `tinker` subdirectory of the present package. Their main purpose is to calculate the molecular mechanics energies and to generate molecular mechanics optimized geometries that may be used in MCSI calculations. To compile these programs, the user should copy the script files from the `tinker/ibmr6k` subdirectory (for IBM/RS6000 computers) or the `tinker/sgi` subdirectory (for SGI computers) into the `tinker/source` subdirectory, as well as the `clock.f` and `calendar.f` source code files. Then, from this source directory, one must launch the compiling scripts in the following order: `compile.make, library.make, link.make`, and `rename.make`. The executables will be located in the `tinker/bin` subdirectory.

VIII. PROGRAM DESCRIPTION

The MCSI program makes use of routines from tinker program through calls from the mct module. Several modified TINKER subroutines are components of the mct module and are used in MCSI instead of the original TINKER subroutines. The version of TINKER we used is version 3.5mn5, which is version 3.5 modified in order to eliminate some bugs and to make the code more suitable for our purposes.

A. Features of TINKER–version 3.5mn5

The MCSI program follows the POLYRATE hooks protocol. This means that if energy, gradient, and/or Hessian calculations are desired, the program calls the appropriate hooks subroutine that will return that information. (The user may read more about hooks in the POLYRATE manual, although everything that is required to be known to use MCSI is given here.) In order to allow the modular hooks protocol, few subroutines of the TINKER program have been modified in order to provide mct module with the requested information and are now considered as being elements of the `mct` module. The subroutines in TINKER that have uses other than the calculation of the energy or first and second derivatives are not used in the MCSI program.

A requirement of MCSI is that the system under study must be defined by means of the symbols of all the chemical elements that compose the system. It is very common in molecular mechanics force fields to represent a group of atoms as a single particle. For example, the methyl group, CH_3, is very often described as a single atom with atomic mass of (approximately) 15 amu, ignoring its internal structure. (This is sometimes called a united-atom representation or a coarse-grained representation.) The present version of MCSI cannot perform such kinds of calculations: the number of atoms in MCSI must be the same as the number of atoms in TINKER, and MCSI only accepts atomic elements. In other words, MCSI can be used with all-atom force fields but not with united-atom force fields.

Another desired feature of MCSI is that the energy and at least its first and second derivatives should be continuous functions of geometry. This is not always fulfilled in some of the force fields in tinker. For example, let's consider the study of butadiene with the MM3 force field. This force field treats the conjugated π-system by means of a self-consistent VESCF calculation, for which no analytical derivatives are available in TINKER, and these derivatives are taken as zero. It is assumed by TINKER–version 3.5 that the user is not going to be interested in points on the potential energy surface far from an equilibrium position, and that the contributions to the overall gradient and Hessian from the derivatives of this term are negligible, but this is not true when the MM energy expression is used as part of an MCSI calculation. Thus, the calculation of a reaction path that involves a modification in the π-system for butadiene will not have continuous derivatives if the contributions to the derivatives from the VECSF calculation are neglected. Therefore, it is our recommendation to never use the VECSF calculation. This is discussed further in Sec. VIII A 3.

With these restrictions, MCSI can use any method available in TINKER for the estimation of energies, gradients, and Hessians. These methods are based on molecular mechanics with a variety of force fields. The user is encouraged to read the TINKER manual for more information about the molecular mechanics methods employed by TINKER. We do warn the reader though that so far we have only checked MCSI with MM3.

A few TINKER subroutines have been modified in order to include some features that are required or useful for following the hooks protocol of MCSI and they are used in MCSI instead of the original TINKER subroutines. In addition, some tinker subroutines have been modified to increase the abilities of tinker program and to fix some bugs, and these changes in the code are presented in more detail in Sec. XIII A. Here we present the modifications that involve changes in the MM methods as implemented in TINKER.

1. *Morse treatment for the stretching terms*

The expression for a Morse curve is:

$$V(R) = D\left(e^{-2\alpha(R-R_o)} - 2\alpha e^{-\alpha(R-R_o)}\right) \qquad (113)$$

where R is the bond distance, R_o is the equilibrium bond distance, α is a parameter (i.e., the range parameter), and D is the bond dissociation energy.

TINKER–version 3.5 includes an option for a Morse treatment of the stretching terms. In version 3.5, the Morse curve is constructed using the bond distance and the force constant read from the parameter file. The α parameter is always set to a default value of 2 Å^{-1}, with the bond dissociation energy estimated from these three parameters. In version 3.5mn3 the user can select the Morse treatment for all bond stretches (as in TINKER–version 3.5) or for only a subset of the bond stretch terms in the molecule. In the latter case, the user of TINKER–3.5mn3 must specify the bond dissociation energy for the bonds to be treated as Morse oscillators and tinker–3.5mn3 maintains the original treatment from the selected force field for the remaining bonds (which usually is a Taylor series expansion). This way, the BONDTYPE keyword is overwritten by MORSE for these bonds, and the default value of the α parameter is also overwritten by the value calculated by the program using bond distance, force constant, and bond dissociation energy.

For treating a bond as a Morse oscillator, the record corresponding to that bond in the parameter file can be followed by the value of the dissociation energy. The BONDTYPE keyword does not need to be set to MORSE; the presence of the bond dissociation energy will overwrite the BONDTYPE keyword for this bond. For example, the following records in the parameter file:

```
BONDTYPE    TAYLOR
bond        1    5       4.7400     1.1120   105.
bond        1    1       4.4900     1.5247
```

will indicate a Morse treatment of the bond between atoms 1 and 5, with the bond dissociation energy for that bond being set to 105 kcal/mol, even though the BONDTYPE option in the parameter file has been set to TAYLOR. Since no bond dissociation energy is input for the 1–1 bond, it will be treated using a Taylor series expansion, as indicated with the BONDTYPE keyword. The records

```
BONDTYPE    MORSE
bond        1    5       4.7400     1.1120   105.
bond        1    1       4.4900     1.5247
```

will lead the program to treat all the bonds as Morse oscillators with the default value of α (2 Å^{-1}), except the 1–5 bond, whose value of α will be estimated according to the bond dissociation energy input.

A new keyword introduced in TINKER is the keyword BDEUNIT. Its syntax is

```
BDEUNIT   real
```

and it allows the user to indicate the scale factor needed to convert the bond dissociation energy input into units of kcal/mol. For example, an alternative way of inputting the same information as in the previous example would be

```
BDEUNIT     627.51
BONDTYPE    MORSE
bond        1    5       4.7400     1.1120   .1673
bond        1    1       4.4900     1.5247
```

In this case, the bond dissociation energy is input in hartrees. The BDEUNIT factor is the coefficient that transforms the bond dissociation energies into units of kcal/mol.

The subroutines modified for introducing this improvement were `analyze.f, ebond.f, ebond1.f, ebond2.f, ebond3.f, field.f, kbond.f, prmkey.f, prtprm.f`, and `readprm.f`. The `bndpot.i, bond.i`, and `kbonds.i` include files were also modified.

In addition to this improvement, a bug that used to cause a miscalculation of the first derivatives of the energy (term `deddt` in `ebond1.f` and `ebond2.f`) when the MORSE option was used has been corrected.

2. MM3 van der Waals term

The original van der Waals term in the MM3 method is represented by the Buckingham potential (also called the Exp-6 potential):

$$V_{\text{Exp-6}}(r) = \epsilon \left[A e^{-Br/r_m} - C \left(\frac{r_m}{r} \right)^6 \right]. \tag{114}$$

where r_m is the sum of the van der Waals radii, and ϵ is the energy parameter for the interaction between two atoms. In TINKER–3.5, the van der Waals energy is represented by either this Exp-6 or by an r^{-12} term, depending on the distance r between the two atoms. If r is larger than a threshold value, the Buckingham function is used; if it is smaller than this threshold, a repulsive function r^{-12} is used instead. The shift between one function and the other is done without any kind of smoothing, making a discontinuous transition from one function to the other. This is not usually a problem in SCMM calculations, since most of the systems that one might study are always in the range described by the Buckingham function, but in multi-configuration molecular mechanics calculations, the system can often reach values of r in the repulsive zone. As stated in Sec. VIII A, we desire the energy to be a continuous function of the coordinates with at least two continuous derivatives. We solved this problem simply by writing the van der Waals term as a linear combination of both functions. Thus, the contribution of the van der Waals term to the energy, V_{vdw}, is written as:

$$V_{\text{vdw}}(r) = \epsilon \left[A e^{-Br/r_m} - C \left(\frac{r_m}{r} \right)^6 \right] + D\alpha \left(\frac{r_m}{r} \right)^{12}, \tag{115}$$

where α is defined as

$$\alpha = \left| \frac{V_{\text{Exp-6}}(r)}{\epsilon (\frac{r_m}{r})^{12}} \right|. \tag{116}$$

and D is a unitless reaction-specific coefficient that can be optimized to minimize the deviation of MCSI energies from single-point accurate energies at a number of points in the dynamically important region (examples are given in Ref. 31). This coefficient can be specified via a keyword `dterm` (which is a keyword that was added in the version 1.1.1 of MCSI) in the `.prm` file that contains the MM3 pa-

rameters. Values of D in the range 0.005-0.02 are expected to work well for most reactions; however, some adjustment may be required for optimum results. Note that all versions of MCSI prior to 1.1.1 and all applications prior to Ref. 31 used a hard-coded value of $D = 0.2$. This resulted in an overestimation of the van der Waals energy at small internuclear separations and in deterioration of the fit (such as appearance of artificial wells on the concave side of the MEP, etc.) that was more or less pronounced in different applications.[3−30] For more details on the van der Waals energy term in MCSI, see Ref. 31

Thus, the MM3 force field now gives a continuous energy with at least two continuous derivatives whenever the VESCF treatment for the π-system is not included. The user should be aware that the results for MM3 calculations obtained with this version of TINKER will be slightly different from those obtained from a standard MM3 code.

The subroutines modified in order to accomplish this objective are `ebuck.f, ebuck1.f, ebuck2.f, ebuck3.f, ebuck4.f, and ebuck5.f`. A bug that miscalculated the second derivatives of the energy for the repulsive term (term d2e in the `ebuck2.f` subroutine) has been corrected in version 1.0.

3. VESCF treatment for π systems

Another relevant issue concerns the VESCF treatment for π-systems, which is mentioned above (in Sec. VIII A). For example, in the 1,3-pentadiene molecule, the bonds 1 and 3 are not a typical double bond, and the bond number 2 is not a typical single bond. Using the same set of parameters for describing the bond numbered 2 and bonds numbered 1 and 3 in pentadiene as any other single and double bond, respectively, leads to an unphysical situation. The VESCF treatment variationally optimizes the parameters of the bonds, bond angles, and torsions involved in a π-system. Thus, after applying the VESCF treatment the bonds number 1 and 3 in 1,3 pentadiene are longer than nonconjugated double bonds, while the bond number 2 is shorter than a single bond located between two single bonds. Similarly, the force constants, bond angles, torsion constants, etc. are optimized for each bond. However, once the new set of parameters describing each interaction is calculated using the VESCF method, the standard MM3 procedure is unsatisfactory for calculating the gradients and Hessians of the systems. In this standard MM3 procedure, if we calculate numerically the derivatives, the parameters will be reoptimized for each geometry, and even though the changes in the geometry are small, they are large enough to make the parameters slightly different for each geometry used in the numerical differentiation. Therefore, the numerical and analytical derivatives are different. (In general, the analytical derivatives are not continuous.)

The way we work around this problem in investigating the isomerization of 1,3-cis-pentadiene[3] is to carry out a VESCF calculation on pentadiene, which is the system that has to be adequately described by the molecular mechanics method. Then, we modify the parameter file, and we define the system using the VESCF-optimized parameters and we do not make any further special treatment for the π-system. This way we obtain for pentadiene exactly the same results as using the VESCF method with the original set of parameters, but a change in the geometry will not modify the parameters, and so we eliminate the problems with the derivatives.

B. Structure of MCSI

The MCSI program follows the hooks protocol in POLYRATE; that is, whenever the program requires energy, gradient, or Hessian calculation, it calls the appropriate hooks subroutines. In this section we will describe all the steps in a usual calculation.

The first subroutine called by the driver is subroutine `mcfope`, which opens the `esp.fu86` output file and the `esp.fu85` input file. If both files are opened succesfully, the next step is to print the header in the `esp.fu86` output file (by calling subroutine `mcpthd`). The program sets the defaults and reads the various options of the calculation (in subroutine `mcrfgi`) and then closes the `esp.fu85` input file (by calling subroutine `mcfclo`). Further, the program prints out a summary table of all the options (in subroutine `mcptab`).

The next step is preparing for the calculation. This is done in subroutine `mcscal`. The first subroutine called in `mcscal` is subroutine `mcsvar`, which sets some variables, followed by subroutine `mcrfmm`. Depending on the type of calculation that is to be pursued (SCMM using configuration 1 bonding scheme, SCMM configuration 2 bonding scheme, or MCSI), the `mcrfmm` subroutine is called to open and read (done by means of subroutine `readfile`) the `esp.fu81` and/or `esp.fu82` input file. The `mcrfmm` subroutine is called with the argument 1 for configuration 1 and 2 for configuration 2. The `esp.fu81` and/or `esp.fu82` input files are `tinker` input files that specify the geometries of the SCMM points (single-configuration molecular mechanics Shepard points), the connectivity, atom types, and atomic symbols of the configuration 1 and configuration 2, respectively. This information (geometries, connectivity, atom types, and atomic symbols) is stored in the appropriate arrays. (Note that for an SCMM calculation, the geometry that is read in the `esp.fu81` or `esp.fu82` input file is meaningless.) For an MCSI calculation, `mcscal` calls three more subroutines: subroutine `mcsici`, which does some checks and determines the internal coordinates used both for Shepard interpolation and for generalized distances, subroutine `mctmmg` or `mctmmgs`, and subroutine `mcsesi`. The `mctmmg` subroutine transforms the geometries of the two SCMM points (from `esp.fu81` and `esp.fu82` input files) from Cartesian to internal coordinates. In case of symmetrized calculations, the subroutine `mcrfmms` is called instead to perform analogous transformations of the $m!$ symmetrically equivalent SCMM geometries that are given in files `esp.fu82` and `esp.fu72-esp.fu76` for configuration I and in files `esp.fu82` and `esp.fu62-esp.fu66` for configuration II (see Sec. IX A for the usage of these files). The `mcsesi` subroutine makes some checks, then calls the `mcrfes` subroutine to read the electronic structure informa-

tion (from `esp.fu83` input file) and the `mctesi` subroutine to transform this information from Cartesian to internal coordinates. The `mctesi` subroutine is presented in more detail in Sec. VIII B 1. Note that the `mcscal` subroutine is called only once prior to any calculation.

After calling the `mcscal` subroutine, MCSI is ready to start the calculation of the energy, gradient, or Hessian, which is done by calling subroutine `mcsrun`. In this subroutine the `esp.fu84` input file is opened, and the number of points for which calculations should be carried out is read. The program then enters a loop (which in the code is a do loop on the integer variable `mcipts`) that runs from 1 to the number of points (`mcnpts`). In this loop, the program reads the geometry of the point, transforms it to mass-unscaled bohrs, calls the appropriate hooks subroutine (`mehook` for an energy calculation, `mghook` for a gradient calculation, or `mhhook` for a Hessian calculation), and then prints the results (in subroutine `mcprez`). In the `mehook` subroutine, the program calls subroutine `mcviic` for an SCMM calculation or subroutines `mcviic` and `mcv12c` for an MCSI calculation. These subroutines (which will be presented in more detail in Secs. VIII B 3 and VIII B 4) return V_{11}, V_{22}, and V_{12}, which are further combined in the `mccene` subroutine to give the potential energy V. Similarly, in the `mghook` and `mhhook` subroutines, the program calls the `mcviic` and/or `mcv12c` subroutines but they return, besides the potential energy components, their first and second derivatives with respect to the Cartesian coordinates, respectively. These derivatives are further combined then in subroutines `mccgra` and `mcches` to produce the first and second derivatives of the potential energy, respectively. Symmetrized MCSI calculations are performed in a similar fashion, except that the subroutines `mehooks`, `mghooks`, and `mhhooks`, respectively, are called to return symmterized MM energies, gradients, and Hessians, respectively, and the `mcv12is` subroutine is called to return a symmetrized value of V_{12}. After the loop over the number of points is completed, the program closes the `esp.fu84` input file and returns to the driver. Finally, the program writes the end of the `esp.fu86` output file, and closes it (by calling subroutine `mcfclo`).

1. *The* `mctesi` *subroutine*

The `mctesi` subroutine is an MCSI subroutine that computes the D, b, and C coefficients of eq. (18) of Ref. 3 for each electronic structure theory point of the Shepard interpolation. This subroutine is called only once since these coefficients are constant.

In this subroutine, after defining some variables, the program enters a loop (which, in the code, is a do loop on the integer variable `mciesp`) that runs from 1 to the number of electronic structure theory points (`mcnesp`). The number of such electronic structure theory points was called $Ns + 1$ in the initial applications of MCSI to chemical reactions.[3,28] For these systems, the electronic structure theory information about the saddle point was required, and all the other electronic structure theory points, Ns, were called "supplementary" points,[3,28] so the total number of electronic structure theory points was therefore

```
mcnesp = Ns + 1
```

In this loop over the number of electronic structure theory points, once for each of these points, the program has to follow the following steps to get the D, b, and C coefficients:

- Transfer the information about each point into the working arrays the store the geometry (Cartesian coordinates) in `x0`, the gradient in `d0`, and the Hessian in `k0`.
- Calculate V_{11} and its gradient and Hessian (that will be stored in `v1`, `d1`, and `k1`, respectively). This is done by calling the mcviic subroutine with the argument 1 (for configuration 1).
- Calculate V_{22} and its gradient and Hessian (that will be stored in `v2`, `d2`, and `k2`, respectively). This is done by calling the `mcviic` subroutine but with the argument 2 (for configuration 2).
- Transform the geometry, gradient, and Hessian from Cartesian coordinates to internal coordinates. (The information we have in `v1`, `v2`, `d1`, `d2`, `k1`, and `k2` is in Cartesian coordinates. This information is obtained from the Cartesian geometry `x0`. Also, the ab initio gradient and Hessian, stored in `d0` and `k0`, are in Cartesian coordinates, and we assume that they correspond to the same orientation as `x0`.) The `mctcic` subroutine is called to transform the Cartesian coordinates `x0` to internal coordinates used for Shepard interpolation and for generalized distances, which are stored in `qs0` and `qd0` respectively. Then the Wilson matrices $\mathbf{B}$ and $\mathbf{G}$ are calculated and applied to `d0`, `d1`, `d2`, `k0`, `k1`, and `k2` to obtain `dxint0`, `dxint1`, `dxint2`, `kxint0`, `kxint1`, and `kxint2`.
- Calculate the differences between gradient and Hessian components used in eqs. (21) and (22) in Ref. 3:

```
do i=1,mcnics
 dxint1(i) = dxint1(i) - dxint0(i)
 dxint2(i) = dxint2(i) - dxint0(i)
    do j=1,mcnics
       kxint1(j,i) = kxint1(j,i) - kxint0(j,i)
       kxint2(j,i) = kxint2(j,i) - kxint0(j,i)
    enddo
enddo
```

- Obtain the D, b, and C coefficients of eq. (18) in Ref. 3 according to eqs. (20) through (22). Those values are stored in the arrays `aevbm(mciesp)`, `bevbm(i,mciesp)` and `cevbm(i,j,mciesp)` where `i` and `j` are indexes of internal coordinates.

After the loop is completed, the program closes the `esp.fu83` input file and leaves the subroutine by returning to subroutine `mcsesi`.

2. *The* `mctesis` *subroutine*

The `mctesis` subroutine is a version of the `mctesi` subroutine that returns D, $\mathbf{b}$, and $\mathbf{C}$ coefficients of eq. (18) of Ref. 3 or, equivalently, of eq. (23) of Ref. 4 for each permutationally equivalent electronic structure theory Shepard point (k,i).

This is accomplished by using the following steps:

- Generate $m!$ values of symmetrically equivalent data points from each Shepard point (k) by applying the permutation operator $\mathbf{P}^{(i)}$ to each $\mathbf{x}^{(k)}$, $\mathbf{G}^{(k)}$ and $\mathbf{F}^{(k)}$ as in eqs. (4-6) of Ref. 4.

- Calculate $m!$ values of V_{11} and their gradients and Hessians (which will be stored in v1_j, d1_j, and k1_j ($j = 1, \cdots m!$), respectively) for each data point (k). This is done by calling the `mcviic` subroutine with the arguments 1 and 1 (for configuration 1 and permutation $\mathbf{P}^{\mathrm{MM}(1)}$ for each geometry (k,i)), and is equivalent to applying each $\mathbf{P}^{\mathrm{MM(j)}}$ to a valence bond configuration 1 at each geometry $(k,1)$.

- Calculate $m!$ values of V_{22} and its gradients and Hessians (which will be stored in `v2_`i, `d2_`i, and `k2_`i ($i = 1, \cdots m!$), respectively) for each data point (k). This is done by calling the `mcviic` subroutine but with the arguments 2 and 1 (for configuration 2 and permutation $\mathbf{P}^{\mathrm{MM}(1)}$ for each geometry (k,i)), and is equivalent to applying each $\mathbf{P}^{\mathrm{MM(j)}}$ to a valence bond configuration 2 at each geometry $(k,1)$.

- Calculate the symmetrized MM potentials $\widetilde{V}_{nn}^{(k)}$ and their gradients $\widetilde{\mathbf{G}}_{nn}^{(k,1)}$ and Hessians $\widetilde{\mathbf{F}}_{nn}^{(k,1)}$ at each point $(k,1)$ using eqs. (10-12) of Ref. 4. These values will be stored in `vn`, `dn`, and `kn` ($n = 1, 2$), for configurations 1 and 2, respectively).

- Generate $m!$ values of $\widetilde{\mathbf{G}}_{nn}^{(k,i)}$ and $\widetilde{\mathbf{F}}_{nn}^{(k,i)}$ from each $\widetilde{\mathbf{G}}_{nn}^{(k,1)}$ and $\widetilde{\mathbf{F}}_{nn}^{(k,1)}$ for each point (k,i) by applying $\mathbf{P}^{(i)}$ to the gradient vectors and Hessian matrices, as in the first step. At this point, we have the symmetrized accurate potential $V^{(k)}$, $\mathbf{G}^{(k,i)}$ and $\mathbf{F}^{(k,i)}$, and the symmetrized MM potential $\widetilde{V}_{nn}^{(k)}$, $\widetilde{\mathbf{G}}_{nn}^{(k,i)}$ and $\widetilde{\mathbf{F}}_{nn}^{(k,i)}$ in Cartesian coordinates at each permutationally equivalent Shepard point.

- The `mctcic` subroutine is further called to transform the geometry of each point (k,i) from Cartesian coordinates to both sets of the internal coordinates $\mathbf{r}$ and $\mathbf{s}$ (*vide supra*). This yields $m!$ sets of coordinates $\mathbf{r}$ and $m!$ sets of coordinates $\mathbf{s}$ (stored in `qs0_`i and `qd0_`i, where `i` corresponds to a permutation $\mathbf{P}^{(i)}$. Then, the Wilson matrices $\mathbf{B}$ and $\mathbf{G}$ are calculated and applied to the symmetrized $\mathbf{G}^{(k,i)}$, $\mathbf{F}^{(k,i)}$, $\widetilde{\mathbf{G}}_{nn}^{(k,i)}$ and $\widetilde{\mathbf{F}}_{nn}^{(k,i)}$ to obtain the differences used in eqs. (20-22) of Ref. 3. These are stored in arrays `dxint0_`i, `dxint1_`i, `dxint2_`i, `kxint0_`i, `kxint1_`i, and `kxint2_`i, where i corresponds to a permutation $\mathbf{P}^{(i)}$. These values are used to calculate the Taylor series coefficients D, $\mathbf{b}$, and $\mathbf{C}$ as in eqs. 20-22 of Ref. 4 at each symmetrically equivalent Shepard point (k,i).

3. *The* `mcviic` *subroutine*

The `mcviic` subroutine is a MCSI subroutine with six arguments: `xii`, `ex`, `gx`, `hx`, `itypec`, and `iconf`. The array `xii` is the array of Cartesian coordinates, `ex` contains the molecular mechanics energy (V_{11} or V_{22}), `gx` array contains the gradient vector of V_{11} or V_{22} with respect to the Cartesian coordinates, and `hx` array contains the Hessian matrix of V_{11} or V_{22} with respect to the Cartesian coordinates. The `itypec` argument is a flag that determines if a molecular mechanics energy (`itypec` = 1), gradient (`itypec` = 2), or Hessian (`itypec` = 3) calculation should be carried out. The `iconf` argument determines whether the molecular mechanics calculation should be carried out using configuration 1 (`iconf` = 1) or configuration 2 (`iconf` = 2).

Another important detail to be kept in mind is that, when a Hessian calculation is desired, the `mcviic` subroutine also has to return the value of the energy and the gradient. The modified TINKER subroutine that carried out molecular mechanics Hessian calculations, the `hesss` subroutine, only returns the value of the Hessian and does not give back the gradient. To determine the gradient and the energy it is necessary to add a call to the `grads` subroutine, also a modified TINKER subroutine, that performs the additional gradient calculation. Similarly, a Hessian calculation has to return the energy; but this is not a problem, since a gradient calculation in TINKER (in the `grads` subroutine) automatically involves an energy calculation.

The `mcviic` subroutine starts by transforming the Cartesian coordinates from bohr (which are the units for coordinates in MCSI) to angstrom (which are the units for coordinates in TINKER). The program then transfers the values of the arrays that store connectivity, atom types, and atomic symbols to the working arrays that are used in the code for the calculations. The values transferred correspond to the configuration 1 or configuration 2 depending on the value of the argument `iconf`. For example, the number of atoms to which an atom is connected is permanently stored in the array `incon`, which is a two-dimensional array, `incon(i,j)`. The index `j` is the value of `iconf` and the index `i` is the number of the atom. For example, in `incon(3,2)` is stored the number of atoms connected to the atom number 3 in the configuration 2 (`iconf` = 2).

After the molecular mechanics information is stored in the working array, the program calls the appropriate modified TINKER subroutines: `energ` for an energy calculation, `grads` for a gradient calculation, or `grads` and `hesss` for a Hessian calculation. Once the molecular mechanics values are returned by TINKER subroutines, the program transforms the energy to hartree and corrects the molecular mechanics energy as described in Sec. II D, eqs. (2.9) and (2.10):

```
ex = dble(enertk)/fhakc - vz1
```

for the configuration 1 and:

```
ex = dble(enertk)/fhakc + ediff0 - vz2
```

for the configuration 2, where `enertk` is the energy in kcal/mol returned by modified tinker subroutines. Finally, the program also transforms the gradient and Hessian to atomic units (hartree bohr^{-1} and hartree bohr^{-2}, respectively) and returns from the subroutine.

4. *The* `mcv12c` *subroutine*

The `mcv12c` subroutine is the part of MCSI that calculates V_{12} and its derivatives in Cartesian coordinates. This subroutine has five arguments: `xii`, `e12`, `g12`, `h12`, and `itypec`. The array `xii` is the array of Cartesian coordinates, `e12` contains the energy V_{12}, `g12` array contains the gradient vector of V_{12} with respect to the Cartesian coordinates, and `h12` array contains the calculated Hessian matrix of V_{12} with respect to the Cartesian coordinates. The `itypec` argument is a flag that indicates which kind of calculation is to be performed: if `itypec` $= 1$, only V_{12} is calculated; if `itypec` $= 2$, the gradient is also calculated; if `itypec` $= 3$, the gradient and Hessian are also calculated.

The `mcv12c` subroutine starts by initializing some variables and doing some checks. If a constant value for V_{12} is wanted (RESMETHOD is set to constant - see Sec. IX E 4), the subroutine returns that V_{12} value. If V_{12} is not constant then the following steps are followed. The subroutine has the geometry that is used to calculate V_{12} and its derivatives stored in the array `xii`. The geometry is transformed to internal coordinates used for Shepard interpolation, that are stored in the array `qs`, and to internal coordinates used for generalized distances, that are stored in the array `qd`. These transformations are done by calling subroutine `mctcic`. If the gradient of V_{12} is to be calculated, it is necessary to obtain the matrix $\mathbf{B}$, which is going to be stored in the array `amatb` in the code. This is done by calling subroutine `ybmat`. If the Hessian of V_{12} is to be calculated, it is necessary to obtain the matrix $\mathbf{G}$, which will be stored in `amatbtc`. The matrix $\mathbf{G}$ is calculated by calling subroutine `btenscj`. After these transformations, it is possible to use the internal coordinates instead of the Cartesian coordinates.

The next step is to calculate V_{12} and its derivatives in internal coordinates by using the appropriate interpolation or expansion scheme. This is the aim of subroutine `mcv12i` (nonsymmetrized calculations), or analogous subroutine `mcv12is` (symmetrized calculations) which is called next. These subroutines (which are explained in Sec. VIII B 5) use the geometry from the arrays `qs` and `qd`, the D, $\mathbf{b}$, and $\mathbf{C}$ coefficients calculated in subroutine mctesi, and determine V_{12} and its derivatives with respect to the internal coordinates either by expansion around one electronic structure theory Shepard point or by Shepard interpolation. The `mcv12i(s)` subroutine then returns the (symmetrized) energy V_{12} in the variable `e12`, its gradient with respect to the internal coordinates in the vector `gi12`, and the Hessian in the array `hi12`.

The last step in subroutine `mcv12c` is to transform the information about the derivatives back to Cartesian coordinates. This is done by calling subroutine `trangj` for getting the gradient in Cartesian coordinates from the gradient in internal coordinates, and by calling the `formf2cj`, `tranlfj`, and `tranfcj` subroutines for getting the Hessian in Cartesian coordinates from the gradient and Hessian in internal coordinates. These transformations are done using the matrix $\mathbf{B}$ calculated previously and stored in the array `amatb`, and the matrix $\mathbf{G}$ stored in `amatbtc`, in order to guarantee that the orientation of these gradient and Hessian is consistent with the orientation of the geometry in Cartesian coordinates from the vector `xii`. Once the term V_{12} and its derivatives in Cartesian coordinates are obtained, the program leaves subroutine `mcv12c` and returns to the appropriate `hooks` subroutine.

5. *The* `mcv12i(s)` *subroutine*

The `mcv12i(s)` subroutine is the subroutine that carries out the Shepard interpolation to obtain V_{12} in internal coordinates as described in Ref. 5. There are two versions of this subroutine: `mcv12i`, which is called in nonsymmetrized calculations, and `mcv12is`, which is called in symmetrized calculations, and they are referred here to as `mcv12i(s)`. The subroutines `mcv12iold` and `mcv12isold` carry out Shepard interpolation of modified Taylor series of V_{12} according to Ref. 3 and they are no longer used in calculations with the MCSI2009 code, unless otherwise is requisted by the user (using the NONHERMITIAN `false` option in the RESONANCE section).

The subroutine `mcv12i(s)` uses the $D^{(k)}$, $\mathbf{b}^{(k)}$, and $\mathbf{C}^{(k)}$ ($D^{(k,i)}$, $\mathbf{b}^{(k,i)}$, and $\mathbf{C}^{(k,i)}$) coefficients calculated in `mctesi(s)` for calculating the value of V_{12} and its derivatives at the geometry `qs` in internal coordinates. This is done using the $m!$`mcnesp` sets of D, $\mathbf{b}$, and $\mathbf{C}$ coefficients. Thus, we obtain $m!$`mcnesp` values of $V_{12}(i)$ and its derivatives. ($V_{12}(k,i)$ represents the value of V_{12} obtained from expansion around Shepard point (k,i)). Usually, we add to this set 2 ($2m!$ in symmetrized calculations) more values: those of the SCMM points. These structures are assumed to be accurately described by the molecular mechanics potential functions therefore their V_{12} and their derivatives are considered to be zero. Once we have the $m!$(`mcnesp`$+2$) values of $V_{12}(i)$, we perform a Shepard interpolation according to eq (2) of Ref. 5 (or equivalently, eq. 88 of Sec. IV for symmetrized calculations) and get the final value of V_{12}. In the `mcv12i(s)old` subroutine, Shepard interpolation if performed according to eq. (14) of Ref. 3 (or equivalently, eq. (24) of Ref. 4 in symmetrized calculations).

Here, `qs` are the internal $\mathbf{r}$ coordinates used for Shepard interpolation, and `qd` are the internal coordinates $\mathbf{s}$ used for generalized distances (see the comment in Sec. II A about the three internal coordinate systems.). The first part of the `mcv12i` subroutine calculates the differences between the values of the internal coordinates at the point described by `qs` and `qd` and the values of the internal coordinates at the Shepard points, `qs0` and `qd0`. These differences are stored in the arrays `dqs` and `dqd`. In the MCSI–1.1.1,

the part of the code that calculates `dqd` was moved to the `mcswts` subroutine.

Next, the `mcswtsnum` subroutine is called to determine the weights and their derivatives with respect to the internal coordinates **s**. The weights involve the generalized distances $dk(\mathbf{s})$ defined by eq. (35) of Ref. 3. The right-hand side of this equation involves adding the squares of coordinates that have different units. Thus eq. (35) is actually evaluated in reduced units. The expression for $d_k(\mathbf{s})$ in reduced units is

$$d_k(\mathbf{s}) = \sqrt{\sum_{i=1}^{N'} \frac{\left(\mathbf{s}_i - \mathbf{s}_i^{(k)}\right)^2}{\mu_i}}, \qquad (117)$$

where $\mu_i \equiv 1a_o$ if s_i is a bond length, and $s_i = 1$ radian if s_i is a bond angle or torsion angle. Since the code uses atomic units, the constants μ_i are all unity, and they do not actually appear in the program.

The next step in the calculation is the differentiation of the weights with respect to the internal coordinates; this is performed numerically, and therefore the final MCSI gradients and Hessians are semi-analytical. The mcswtsnum subroutine uses the step size stored in a variable hh to generate coordinates for evaluating numerical derivatives, and it calls either the mcswts or mcswtss subroutine (depending on whether one performes a non-symmetrized or symmetrized MCSI calculation) that return the normalized weights. In the `mcswts/mcswts` subroutine, taking the differences between the internal coordinates used for calculating generalized distances (stored in `dqd/dqd_p12`), the program calculates the $m!$(`mcnesp + 2`) generalized distances according to eq. (35) of Ref. 3 or, equivalently, according to eq. (38) of Ref. 4, and stores them in the array `disst(i)`. In order to avoid problems with the code, we check that none of these distances are smaller than a cutoff. If one of the distances is less than that cutoff, we consider that the weight of that point in the Shepard interpolation is unity, while the weight of all the other points is zero. If all the generalized distances are larger than the cutoff, we calculated the normalized weights given in eq. (34) of Ref. 3 and their numerical derivatives with respect to the internal coordinates **s**. (The cutoff is currently hardwired and set equal to 10-8 reduced units.) Note that the code is written in such a way that $w(i)$ is the array that contains the normalized weights of each Shepard point. Testing new weight functions will only require to code the new mathematical functions that evaluate the values of the components of the array $w(i)$. Therefore the code, as it is now, is quite modular and easy to modify.

Once the weights and their derivative are calculated, the program starts a loop over the Shepard points, in order to evaluate $V_{12}(i)$, which is stored as `e12pnt (e12pnt_i)`, by using the arrays `aevbm`, `bevbm_`i, and `cevbm_`i calculated in subroutine `mctesi(s)`, and eq. (18) of Ref. 3. The contribution of this Shepard point to the interpolated value of V_{12}, stored in `e12`, is calculated as:

```
e12 = e12 + w(mciesp)*e12p(mciesp)
```

The program also calculates the gradient of V_{12} for the present Shepard point (in internal coordinates) and stores it in `gi12`, and the same thing is done with the Hessian matrix, which is stored in `hi12`. The loop over the Shepard points is closed here, and after the loop has run over the m!mcnesp electronic structure theory Shepard points, the interpolated V_{12} is passed to `mcv12c` through the `e12` variable, its gradient passed through `gi12`, and the Hessian through `hi12`.

C. Limitations on the use of MCSI

There are some issues that limit the use of MCSI:

- The MCSI method in its current state can make use of only two molecular mechanics configurations. This is a formal limitation, but we note that the results still converge to the correct answer as the amount of electronic structure information is increased.

- In addition to the geometries where electronic structure information is available, only two more structures can be used for the Shepard interpolation step. These two structures (called SCMM points) are points where the single-configuration molecular mechanics potentials defined by configuration 1 and configuration 2, respectively, are accurate. For example, the SCMM point for configuration 1 can be a single reactant (for unimolecular reactions) or an optimized reactant well (for bimolecular reactions), and the SCMM point for configuration 2 can be a single product (for unimolecular reactions) or a optimized product well (for bimolecular reactions).

- The ordering of the atoms has to be the same for the SCMM points, for the electronic structure theory Shepard points, and in defining the other points where an MCSI calculation is desired.

- For symmetrized MCSI calculations, the identical nuclei that need to be treated as indistinguishable, should have atom numbers 2, 3, and 4 when defining Cartesian coordinates in all input files (see, e.g., test runs test13 and test14).

- All the atoms of the system under study must be included explicitly. It is very common in molecular mechanics force fields to represent a group of atoms as a single particle. For example, the methyl group, CH_3, it is very often described as a single atom with atomic mass of (approximately) 15 amu, ignoring its internal structure. This is sometimes called united-atom coarse-grained representation. The present version of MCSI cannot perform such kinds of calculations; the number of atoms in MCSI (or the electronic structure package) has to be the same as the number of atoms in TINKER, and MCSI (or the electronic structure package) only accepts atomic elements.

IX. DESCRIPTION OF INPUT FILES

A. File usage

The MCSI program uses several files for input and output data. The names of these files are of the form `esp.fu#`, where # is an integer indicating the FORTRAN unit number. The usage of the files is:

Input files:

- `esp.fu81` Input data for the configuration 1: the geometry of the SCMM point for configuration 1, the atom types, and the connectivities
- `esp.fu82` Input data for the configuration 2: the geometry of the SCMM point for configuration 2, the atom types, and the connectivities
- `esp.fu83` Input data for the electronic structure theory Shepard points used in Shepard interpolation (geometry, energy, gradient, Hessian)
- `esp.fu84` Input data for the points (the number and the geometries of the points) where MCSI (SCMM) calculations are desired
- `esp.fu85` General input data that controls the MCSI or SCMM part of the calculation
- `param.prm` Parameter file for the molecular mechanics force field (FORTRAN unit 79) required by tinker.

A more detailed description of the input files is presented in the following sections.

In addition, for symmetrized calculations with $m = 3$, the following input files are required:

- `esp.fu72` Input data for the configuration 1: the geometry of the SCMM point for configuration 1 is the same as in `esp.fu81` file, but with permuted atom types and connectivities. These atom types and connectivities result from applying $\mathbf{P}^{\mathrm{MM}(12)}$ to the atom types and connectivities specified in `esp.fu81`
- `esp.fu73` Same as `esp.fu72` but applying $\mathbf{P}^{\mathrm{MM}(23)}$.
- `esp.fu74` Same as `esp.fu72` but applying $\mathbf{P}^{\mathrm{MM}(13)}$.
- `esp.fu75` Same as `esp.fu72` but applying $\mathbf{P}^{\mathrm{MM}(123)}$.
- `esp.fu76` Same as `esp.fu72` but applying $\mathbf{P}^{\mathrm{MM}(132)}$.
- `esp.fu62` Input data for configuration 2: the geometry of the SCMM point for configuration 1 is the same as in `esp.fu82` file, but with permuted atom types and connectivities. These atom types and connectivities result from applying $\mathbf{P}^{\mathrm{MM}(12)}$ to the atom types and connectivities specified in `esp.fu82`.
- `esp.fu63` Same as `esp.fu62` but applying $\mathbf{P}^{\mathrm{MM}(23)}$.
- `esp.fu64` Same as `esp.fu62` but applying $\mathbf{P}^{\mathrm{MM}(13)}$.
- `esp.fu65` Same as `esp.fu62` but applying $\mathbf{P}^{\mathrm{MM}(123)}$.
- `esp.fu66` Same as `esp.fu62` but applying $\mathbf{P}^{\mathrm{MM}(132)}$.
- Output files: `esp.fu86` Full output file

B. Description of `esp.fu81` and `esp.fu82` input files

The esp.fu81 and `esp.fu82` input files contain the geometries of the single-configuration molecular mechanics Shepard points (the SCMM point for configuration 1 and the SCMM point for configuration 2, respectively), the atom types, and the connectivities for the configurations 1 and 2, respectively. These input files are written in the same fashion as any xyz-input file for TINKER. The only consideration to be taken into account is the restriction on the ordering of atoms, namely, the order used in these files should be the same as the order used for MCATOMS keywords in MCGENERAL section of the `esp.fu85` input file. For the symmetrized MCSI calculations, additional input files `esp.fu62-esp.fu66` and `esp.fu72-esp.fu76` are required. These files specify the corresponding permuted MM configurations. An example is given in the `test14` directory.

The input file has the following format: On the first record, there must be an integer indicating the number of atoms in the system that may be followed by some comments. The following records contain the information on these atoms: a number indicating the index of the atom, the atomic symbol of the atom, the three Cartesian coordinates, the atom type number, and the indices of the atoms bonded to it. All the records are read in free format.

Example:

```
7  HOH---CH3 well
1  H   -2.528735  0.393818 -0.630578   5  2
2  C*  -2.225690 -0.619356 -0.324313  29  1  4  5
3  H     .457798  1.026333  0.821507  21  6
4  H   -1.822563 -1.322295 -1.069648   5  2
5  H   -2.320781 -0.927796  0.728296   5  2
6  O    1.068112  0.590172  0.243503   6  3  7
7  H    1.013893 -0.326197  0.476171  21  6
```

In addition, for an EE-MM or EE-MCSI calculations, the CRKFMT and CRK sections are required after the xyz input for TINKER. The CRK section specifies partial atomic charges (this is called the charge distribution), electrostatic potentials at the atomic centers (this is called the electrostatic potential distribution), derivatives of charges with respect to coordinates, $\kappa_{ab}^{(i)} \equiv \dfrac{\partial Q_a^{(i)}}{\partial R_b}$, and derivatives of charges with respect to electrostatic potentials, $\chi_{ab}^{(i)} \equiv \dfrac{\partial Q_a^{(i)}}{\partial \Phi_b}$ for the diagonal elements (eq. 19). The CRKFMT section specifies the format and units. Note that the matrix $\boldsymbol{\kappa}^{(i)}$ is read in terms of the Cartesian coordinates, then it is transformed to the internal coordinates, which are defined in ICSHEPARD in the RESONANCE section in the `esp.fu85 file`, in an EE-MM or EE-MCSI calculation. The table below lists all valid keywords for these sections. A more complete explanation for each keyword is given in Secs. IX C 2 and IX C 3.

The CRKFMT section

(This section is similar to the EEGEN83 one in the `esp.fu83` file.)

Keyword	Type	Default	Description
FORMQPHI	variable	*1*	the format for the charges and electrostatic potentials
UNITELP	variable	*volt*	the units for the electrostatic potentials
FORMDQDR	variable	*gamess*	the format for the derivatives of the charges with respect to coordinates
UNITDQDR	variable	*au*	the units for the derivatives of the charges with respect to coordinates
FORMDQDPHI	variable	*gamess*	the format for the derivatives of the charges with respect to electrostatic potentials
UNITDQDPHI	variable	*au*	the units for the derivatives of the charges with respect to the electrostatic potentials

The CRK section

(This section is similar to the POINT one in the `esp.fu83` file.)

Keyword	Type	Default	Description
QPHI	list	required	charges and electrostatic potentials for the diagonal elements
DQDR	list	required	derivatives of charges with respect to coordinates for the diagonal elements
DQDPHI	list	required	derivatives of charges with respect to electrostatic potentials for the diagonal elements

Example of `esp.fu81` *or* `esp.fu82` *file for an EE-MM or EE-MCSI calculation:*

```
     6  Cl...CH3Cl
     1  C      0.000000    0.000000    0.190194     1     2     3     4     5
     2  H     -0.512622    0.887887   -0.157990     5     1
     3  H     -0.512622   -0.887887   -0.157990     5     1
     4  H      1.025244    0.000000   -0.157990     5     1
     5  Cl     0.000000    0.000000    1.962454    12     1
     6  Cl     0.000000    0.000000  -10.000000   200

*CRKFMT

UNITELP au
UNITDQDPHI au
UNITDQDR bohr
FORMQPHI 2

*CRK

QPHI
 1 -0.1697892085E+00  0.0000000000E+00
 2  0.1142936768E+00  0.0000000000E+00
 3  0.1142936768E+00  0.0000000000E+00
 4  0.1142936768E+00  0.0000000000E+00
 5 -0.1730918219E+00  0.0000000000E+00
 6 -0.1000000000E+01  0.0000000000E+00
END

DQDR
 1  1 0.00000000E+00 0.00000000E+00-2.27867592E-01 5.94803140E-03-1.03022926E-02
 1  2 1.75759847E-02 5.94803140E-03 1.03022926E-02 1.75759847E-02-1.18960628E-02
 1  3 0.00000000E+00 1.75759847E-02 0.00000000E+00 0.00000000E+00 1.75139638E-01
 1  4 0.00000000E+00 0.00000000E+00 0.00000000E+00
 2  1-3.12183735E-03 5.40718090E-03-3.17427793E-02-1.26748426E-02 2.19534713E-02
 2  2-5.69980378E-04 1.15642832E-02-9.85694039E-03 4.72853869E-03 2.75421917E-03
 2  3-1.49434332E-02 4.72853869E-03 1.78125596E-03-3.08522583E-03 2.28556823E-02
 2  4 0.00000000E+00 0.00000000E+00 0.00000000E+00
```

```
 3  1-3.12183735E-03-5.40718090E-03-3.17427793E-02 1.15642832E-02 9.85694039E-03
 3  2 4.72853869E-03-1.26748426E-02-2.19534713E-02-5.69980378E-04 2.75421917E-03
 3  3 1.49434332E-02 4.72853869E-03 1.78125596E-03 3.08522583E-03 2.28556823E-02
 3  4 0.00000000E+00 0.00000000E+00 0.00000000E+00
 4  1 6.24367469E-03 0.00000000E+00-3.17427793E-02-1.43185024E-02-5.08649285E-03
 4  2 4.72853869E-03-1.43185024E-02 5.08649285E-03 4.72853869E-03 2.53496852E-02
 4  3 0.00000000E+00-5.69980378E-04-3.56251193E-03 0.00000000E+00 2.28556823E-02
 4  4 0.00000000E+00 0.00000000E+00 0.00000000E+00
 5  1 0.00000000E+00 0.00000000E+00 3.23095931E-01 9.48103137E-03-1.64216280E-02
 5  2-2.64630823E-02 9.48103137E-03 1.64216280E-02-2.64630823E-02-1.89620627E-02
 5  3 0.00000000E+00-2.64630823E-02 0.00000000E+00 0.00000000E+00-2.43706684E-01
 5  4 0.00000000E+00 0.00000000E+00 0.00000000E+00
 6  1 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
 6  2 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
 6  3 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
 6  4 0.00000000E+00 0.00000000E+00 0.00000000E+00
END

DQDPHI
 1  1-3.20965502E+00 7.27741772E-01 7.27741772E-01 7.27741772E-01 1.02642970E+00
 1  2 0.00000000E+00
 2  1 7.27741772E-01-9.41489938E-01 4.07115495E-02 4.07115495E-02 1.32325067E-01
 2  2 0.00000000E+00
 3  1 7.27741772E-01 4.07115495E-02-9.41489938E-01 4.07115495E-02 1.32325067E-01
 3  2 0.00000000E+00
 4  1 7.27741772E-01 4.07115495E-02 4.07115495E-02-9.41489938E-01 1.32325067E-01
 4  2 0.00000000E+00
 5  1 1.02642970E+00 1.32325067E-01 1.32325067E-01 1.32325067E-01-1.42340490E+00
 5  2 0.00000000E+00
 6  1 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
 6  2 0.00000000E+00
END
```

C. Description of `esp.fu83` input file

The input for FORTRAN unit `esp.fu83` is in the free format keyword style. The input has the same for both symmetrized and unsymmetrized calculations, and it consists of section headers (section headers are always preceded by a star, *) and associated keywords. The esp.fu83 input consists of two types of sections: MCGEN83, which gives general information about the file, and POINT, which give the electronic structure data for an electronic structure theory Shepard point. Note the sections can be entered in any order in the input file.

The keywords used in this file are of two types: variable and list. A variable keyword always requires an argument. The argument must appear on the same line as the variable keyword. The second type is the list keyword. The use of a list keyword used here takes the form:

```
LISTKEYWORD
...
...
END
```

where the END line is mandatory.

All input is case insensitive, and anything after a pound sign (#) on a line is assumed to be a comment. Blank lines are ignored. Following an example for the `esp.fu83` input file, the rest of this section details the keywords for the two sections. Please note that every keyword must be used in its proper section, i.e., between the occurrence of its starred header and the occurrence of the next starred header (or, for the last section, between the proper starred header and the end of the file).

Example:

```
*MCGEN83
NESTSP 3

*POINT 1
ENERGY 0.0189494
```

```
GEOMETRY
 -6.69454329E-02 -1.96574211E+00  0.00000000E+00 -9.15479645E-01  8.85833624E-01
  0.00000000E+00 -1.03439821E-01  3.77243731E+00  0.00000000E+00
END
GRADIENT
 -2.80934625E-08 -7.33177670E-09  2.12772599E-11 -1.41410517E-08  3.67943859E-08
  1.46347707E-11  4.22345150E-08 -2.94626079E-08 -3.59120307E-11
END
HESSIAN
  7.26849015E-03 -2.46048400E-02  9.35414373E-02 -4.14090454E-12  5.92075782E-12
 -1.80332522E-08 -1.58285758E-02  5.21279047E-02 -6.41965167E-12  2.30461818E-02
 -6.01027280E-03  1.56393041E-02 -4.61939022E-11 -2.75845692E-02 -1.39825157E-01
  1.52073611E-11 -3.72014964E-11 -2.99670799E-08  1.54332795E-11  1.38812605E-10
  8.11309802E-08  8.56008545E-03 -2.75230647E-02  1.05605569E-11 -7.21760599E-03
  3.35948420E-02 -3.06406234E-11 -1.34247944E-03  3.06151128E-02 -1.09180741E-01
  4.02730797E-11 -2.45433355E-02  1.24185853E-01 -1.01611061E-10 -6.07177726E-03
 -1.50051120E-02 -1.10664188E-11  3.12807014E-11  4.74362771E-08 -9.01366720E-12
 -9.26186406E-11 -5.11638840E-08  2.00800682E-11  6.13379559E-11  3.77438791E-09
END

*POINT 2
ENERGY 0.0155076
GEOMETRY
  1.88783626E-02 -1.77342847E+00  0.00000000E+00 -9.64608741E-01  7.58182633E-01
  0.00000000E+00 -1.48060031E-02  3.98045268E+00  0.00000000E+00
END
GRADIENT
 -7.04527640E-03  2.08625864E-02  4.20704162E-11  1.23914913E-02  9.85972298E-05
 -1.53427741E-11 -5.34621492E-03 -2.09611837E-02 -2.67276420E-11
END
HESSIAN
  2.73425524E-02 -8.91295912E-02  2.38375655E-01 -1.64800680E-11  3.20686826E-11
 -7.62842906E-03 -3.22699724E-02  9.74643152E-02  1.37702563E-11  3.31359082E-02
  7.49672027E-02 -1.93418930E-01 -3.23699396E-11 -7.95148555E-02  1.64483703E-01
  2.15023069E-11 -2.75062061E-11  7.14722292E-03 -1.28980457E-11  6.75315374E-11
 -1.27930503E-02  4.92742148E-03 -8.33472403E-03  2.70970039E-12 -8.65935818E-04
  4.54765281E-03 -8.60421477E-12 -4.06148564E-03  1.41623885E-02 -4.49567246E-02
  3.01891512E-13 -1.79494597E-02  2.89352272E-02 -4.00256698E-11  3.78707122E-03
  1.60214974E-02 -5.02228204E-12 -4.56263118E-12  4.81206933E-04 -8.72237563E-13
 -3.51617696E-11  5.64582739E-03  5.89458449E-12  3.97241047E-11 -6.12703432E-03
END

*POINT 3
ENERGY 0.0131228
GEOMETRY
  1.87914352E-02 -1.77251762E+00  0.00000000E+00 -7.94928355E-01  1.33468323E+00
  0.00000000E+00 -1.94981928E-02  3.96178219E+00  0.00000000E+00
END
GRADIENT
 -5.58392215E-03  1.98776164E-02  2.12149634E-11 -2.66994512E-04 -4.12160102E-02
 -2.25140647E-10  5.85091666E-03  2.13383939E-02  2.03925684E-10
END
HESSIAN
 -4.59086210E-03 -4.15311460E-03  2.00441257E-02  1.03647292E-11 -2.58418012E-10
 -6.61590357E-03 -2.57695064E-03  2.90930715E-02  8.79571777E-11  2.58958679E-02
 -1.72503621E-02  7.60444963E-02  3.56457266E-10  1.69607045E-02 -6.60072201E-02
 -4.80443485E-11  6.15093126E-10  6.87432965E-03 -1.89563233E-10 -9.00231123E-10
  6.83804279E-04  7.16781445E-03 -2.49399568E-02 -9.83219024E-11 -2.33189172E-02
  2.89657547E-04  2.37607569E-10  1.61511028E-02  2.14034767E-02 -9.60886206E-02
 -9.80392519E-11 -4.60537760E-02 -1.00372762E-02  2.85137984E-10  2.46502993E-02
```

```
  1.06125897E-01  3.76796288E-11 -3.56675142E-10 -2.58425284E-04  1.01606046E-10
  5.43773903E-10 -7.55813393E-03 -1.39285666E-10 -1.87098750E-10  7.81655924E-03
END
```

1. The MCGEN83 section

The MCGEN83 section is used to give the number of electronic structure theory points to be used in the Shepard interpolation, the units of the electronic structure data, and the format of the Hessian matrices. The table below lists all valid keywords for this section. A more complete explanation for each keyword is given in the alphabetical glossary that follows. Note that all keywords are variable keywords, and they all have default values. These keywords need to be given only if the user wishes to override the default.

Keyword	Type	Default	Description
FORMHESS	variable	*packed*	specify the format for the Hessians given in the `esp.fu83` file
NESTSP	variable	1	number of electronic structure theory Shepard points
UNITENER	variable	*hartree*	specify the units for the energies given in the `esp.fu83` file
UNITGEOM	variable	*bohr*	specify the units for the geometries given in the `esp.fu83` file
UNITGRAD	variable	*hperb*	specify the units for the gradients given in the `esp.fu83` file
UNITHESS	variable	*hperb2*	specify the units for the Hessians given in the `esp.fu83` file

Glossary of MCGEN83 keywords

FORMHESS

FORMHESS is a variable keyword that specifies the format in which the Hessian matrices are given. The options are the packed format (as printed in a Gaussian formatted checkpoint file), the full format, or the GAMESS format. The default is `packed`.

Setting FORMHESS to 'ZERO' requests using no Hessian information in MCSI calculations as described in Ref. 8 This is equivalent to truncation of the Taylor series at the first order.

The packed format of the Hessian matrix should list only the lower triangle of the matrix in order such that $H_{ij} = F(j + i(i-1)/2)$, where $F(k)$ is the k element of the input list and H_{ij} is the element of matrix **H** in row i and column j. The number of elements listed per line is arbitrary. Examples of both input forms for a 6×6 matrix are shown below.

Example 1 of full matrix:

```
H11 H12 H13 H14 H15 H16
H21 H22 H23 H24 H25 H26
H31 H32 H33 H34 H35 H36
H41 H42 H43 H44 H45 H46
H51 H52 H53 H54 H55 H56
H61 H62 H63 H64 H65 H66
```

Example 2 of full matrix:

```
H11 H12 H13 H14 H15 H16 H21 H22 H23
H24 H25 H26 H31 H32 H33 H34 H35 H36
H41 H42 H43 H44 H45 H46 H51 H52 H53
H54 H55 H56 H61 H62 H63 H64 H65 H66
```

Example 1 of packed matrix:

```
H11
H21 H22
H31 H32 H33
H41 H42 H43 H44
H51 H52 H53 H54 H55
H61 H62 H63 H64 H65 H66
```

Example 2 of packed matrix:

```
H11 H21 H22 H31 H32
H33 H41 H42 H43 H44
H51 H52 H53 H54 H55
H61 H62 H63 H64 H65
H66
```

Example of GAMESS matrix:

```
 1  1 5.64593548E-05 0.00000000E+00-3.25451560E-04-5.64593548E-05 0.00000000E+00
 1  2 3.25451560E-04
```

```
2  1 0.00000000E+00 5.64593340E-05-3.25449030E-04 0.00000000E+00-5.64593340E-05
2  2 3.25449030E-04
3  1-3.25451560E-04-3.25449030E-04 3.28251414E-01-3.25451561E-04-3.25449031E-04
3  2-3.23469728E-01
4  1-5.64593548E-05 0.00000000E+00-3.25451561E-04 5.64593548E-05 0.00000000E+00
4  2 3.25451561E-04
5  1 0.00000000E+00-5.64593340E-05-3.25449031E-04 0.00000000E+00 5.64593340E-05
5  2 3.25449031E-04
6  1 3.25451560E-04 3.25449030E-04-3.23469728E-01 3.25451561E-04 3.25449031E-04
6  2 3.18688043E-01
```

Note that *Gaussian 98* and *Gaussian 03* write the packed Hessian matrix in a slightly different way (that is, different from GAMESS) in the standard output file, but the Hessian in the formatted checkpoint file (requested by the FormCheck or FChk keyword) is in precisely the packed form accepted by MCSI and so is convenient for copying and pasting into MCSI `esp.fu83` files.

Note that the exact **H** matrix is symmetric, but due to finite precision in computations, the actual matrix one has to input is usually unsymmetric. If one inputs an unsymmetric full matrix, the code uses the unsymmetric matrix. If a user wants the code to use a symmetric matrix, we recommend taking an arithmetic average of H_{ij} and H_{ji} for the resulting symmetrized elements in packed form.

Option	Description
packed	packed Hessian matrices are given
full	full Hessian matrices are given
gamess	the GAMESS format is used for the Hessian matrices
zero	no Hessian information is used

Example:

```
FORMHESS full
```

NESTSP

NESTSP is a variable keyword that specifies the number of electronic structure theory Shepard points that are given in the `esp.fu83` input file and are used in the Shepard interpolation of resonance function. The default is `1`.

Example:

```
NESTSP 10
```

UNITENER

UNITENER is a variable keyword that specifies the units for the energies given in the POINT sections of the `esp.fu83` input file. The default is `hartree`.

Option	Description
hartree	Energies are given in hartrees (atomic units)
kcal	Energies are given in kcal/mol

Example:

```
UNITENER kcal
```

UNITGEOM

UNITGEOM is a variable keyword that specifies the units for the geometries given in the POINT sections of the `esp.fu83` input file. The default is `bohr`.

Option	Description
ang	Geometries are given in angstroms
bohr	Geometries are given in bohrs (atomic units)

Example:

```
UNITGEOM ang
```

UNITGRAD

UNITGRAD is a variable keyword that specifies the units for the gradients given in the POINT sections of the `esp.fu83` input file. The default is `hperb`.

Option	Description
hperb	Gradients are given in hartree bohr^{-1} (atomic units)
kcpera	Gradients are given in kcal mol^{-1} angstrom^{-1}

Example:

```
UNITGRAD kcpera
```

UNITHESS

UNITHESS is a variable keyword that specifies the units for the Hessians given in the POINT section of the `esp.fu83` input file. The default is `hperb2`.

Option	Description
hperb2	Gradients are given in hartree bohr^{-2} (atomic units)
kcpera2	Gradients are given in kcal mol^{-1} angstrom^{-2}

Example:

```
UNITHESS kcpera2
```

2. The EEGEN83 section

The EEGEN83 section is used to give the units and formats of the electronic structure data used as the electronic-structure-theory Shepard points in an EE-MCSI calculation. Note that the electronic structure data in an EE-MM calculation or for the diagonal elements in an EE-MCSI calculation is read in the CRK section in the `esp.fu81` or `esp.fu82` files (See Sec. IX B). This section is read in when EECALC in the EE section in the `esp.fu85` file is true. The table below lists all valid keywords for this section. A more complete explanation for each keyword is given in the alphabetical glossary that follows. Note that all keywords are variable keywords, and they all have default values. These keywords need to be given only if the user wishes to override the default.

Keyword	Type	Default	Description
FORMQPHI	variable	1	the format for the charges and electrostatic potentials given in the `esp.fu83` file
UNITELP	variable	*volt*	the units for the electrostatic potentials given in the `esp.fu83` file
FORMDQDR	variable	*gamess*	the format for the derivatives of the charges with respect to coordinates given in the `esp.fu83` file
UNITDQDR	variable	*au*	the units for the derivatives of the charges with respect to coordinates given in the `esp.fu83` file
FORMDQDPHI	variable	*gamess*	the format for the derivatives of the charges with respect to electrostatic potentials given in the `esp.fu83` file
UNITDQDPHI	variable	*au*	the units for the derivatives of the charges with respect to the electrostatic potentials given in the `esp.fu83` file

Glossary of EEGEN83 keywords

FORMDQDPHI

FORMDQDPHI is a variable keyword that specifies the format in which the matrices representing the derivatives of the charges with respect to electrostatic potentials, $\chi_{ab} \equiv \dfrac{\partial Q_a}{\partial \Phi_b} = \dfrac{\partial^2 V^{\mathrm{EEQM}}}{\partial \Phi_a \partial \Phi_b}$, are given in the `esp.fu83` file. The options are the same as FORMHESS in the MCGEN83 section: the packed format (as printed in a *Gaussian* formatted checkpoint file), the full format, or the GAMESS format. Setting FORMDQDPHI to 'ZERO' requests using no χ_{ab} information in EE-MCSI calculations as the case of the gradient-based MCSI. The default is `gamess`.

Option	Description
packed	packed matrices are given
full	full matrices are given
gamess	the GAMESS format is used for matrices
zero	no χ_{ab} is used

Example:

```
FORMDQDPHI full
```

FORMDQDR

FORMDQDR is a variable keyword that specifies the format in which the matrices representing the derivatives of the charges with respect to coordinates, $\kappa_{ab} \equiv \dfrac{\partial Q_a}{\partial R_b} = \dfrac{\partial^2 V^{\mathrm{EEQM}}}{\partial \Phi_a \partial R_b}$, are given in the `esp.fu83` file. The options are the full format or the GAMESS format. Note that the packed format is unavailable because $\boldsymbol{\kappa}$ is not symmetric. Setting FORMDQDR to 'ZERO' requests using no κ_{ab} information in EE-MCSI calculations as the case of the gradient-based MCSI. The default is `gamess`.

Option	Description
full	full matrices are given
gamess	the GAMESS format is used for matrices
zero	no κ_{ab} is used

Example:

```
FORMDQDPHI full
```

FORMQPHI

FORMQPHI is a variable keyword that specifies the format in which the partial charges and electrostatic potentials are given in the `esp.fu83` file. The ordering of atoms must be the same as that of MCATOMS in the MCGENERAL section in the `esp.fu85` file (See Sec. IX E 1). The

options are two formats, format 1 or format 2 (see examples). The default is `1`.

Option	Description
1	*mcnatm* lines, each indicating the charge and the electrostatic potential on a given atom.
2	*mcnatm* lines, each of which has a comment field (the first word of each line is ignored in a calculation), followed by the charges, and the electrostatic potentials on a given atom.

Example:

```
FORMQPHI 2
```

Example of format 1:

```
QPHI
 -0.2600038533E-01  0.0000000000E+00
  0.1185141676E+00  0.0000000000E+00
  0.1185141676E+00  0.0000000000E+00
  0.1185141676E+00  0.0000000000E+00
 -0.6647710587E+00  0.0000000000E+00
 -0.6647710587E+00  0.0000000000E+00
END
```

Example of format 2:

```
QPHI
 1 -0.2600038533E-01  0.0000000000E+00
 2  0.1185141676E+00  0.0000000000E+00
 3  0.1185141676E+00  0.0000000000E+00
 4  0.1185141676E+00  0.0000000000E+00
 5 -0.6647710587E+00  0.0000000000E+00
 6 -0.6647710587E+00  0.0000000000E+00
END
```

UNITDQDPHI

UNITDQDPHI is a variable keyword that specifies the units for the derivatives of the charges with respect to electrostatic potentials, $\chi_{ab} \equiv \frac{\partial Q_a}{\partial \Phi_b} = \frac{\partial^2 V^{\mathrm{EEQM}}}{\partial \Phi_a \partial \Phi_b}$, given in the POINT sections of the `esp.fu83` input file. The default is `au`.

Option	Description
au	χ are given in atomic units
volt	χ are given in e/V

Example:

```
UNITDQDPHI volt
```

UNITDQDR

UNITDQDR is a variable keyword that specifies the units for the derivatives of the charges with respect to coordinates, $\kappa_{ab} \equiv \frac{\partial Q_a}{\partial R_b} = \frac{\partial^2 V^{\mathrm{EEQM}}}{\partial \Phi_a \partial R_b}$, given in the POINT sections of the `esp.fu83` input file. The default is `au`.

Option	Description
au	χ are given in atomic units
ang	χ are given in $e/\mathring{A}$

Example:

```
UNITDQDR ang
```

UNITELP

UNITELP is a variable keyword that specifies the units for the electrostatic potentials, given in the POINT sections of the `esp.fu83` input file. The default is `volt`.

Option	Description
au	electrostatic potentials are given in atomic units
volt	electrostatic potentials are given in volts

Example:

```
UNITDQDR volt
```

3. The POINT sections

The `esp.fu83` input file will have a number of POINT sections that should equal the number of electronic structure theory points to be used in Shepard interpolation. Each POINT section is used to give the electronic structure information for one of the electronic structure theory Shepard points. Each of the POINT sections is recognized by a line containing *POINT followed by an integer, which represent the index of the electronic structure theory Shepard point. The presence of this integer is required. The table below lists all valid keywords for this section. A more complete explanation for each keyword is given in the alphabetical glossary that follows. Note that all keywords are required, and none of them have default values.

Keyword	Type	Default	Description
			The first four parameters are required for either MCSI or EE-MCSI:
ENERGY	variable	required	energy of the electronic structure theory Shepard point
GEOMETRY	list	required	geometry of the electronic structure theory Shepard point
GRADIENT	list	required	gradient components of the electronic structure theory Shepard point
HESSIAN	list	required	Hessian components of the electronic structure theory Shepard point
			The following two parameters are optional:
ICSHZERO	list	none	internal coordinates whose quadratic Taylor coefficients are set equal to zero
ICSAZERO	list	none	internal coordinates whose linear and quadratic Taylor coefficients are set equal to zero
			The following parameters are required only for an EE-MCSI calculation:
QPHI	list	none	charges and electrostatic potentials of the electronic structure theory Shepard point
DQDR	list	none	derivatives of charges with respect to coordinates of the electronic structure theory Shepard point
DQDPHI	list	none	derivatives of charges with respect to electrostatic potentails of the electronic structure theory Shepard point

Glossary of POINT keywords

ENERGY

ENERGY is a variable keyword that specifies the energy of the electronic structure theory Shepard point. Note that this energy should be relative to the energy of the configuration 1 energy reference state as presented in Sec. II D of this manual.

Example:

```
ENERGY 0.0189494
```

GEOMETRY

GEOMETRY is a list keyword that specifies the geometry of the electronic structure theory Shepard point.

Example:

```
GEOMETRY
-6.69454329E-02 -1.96574211E+00  0.00000000E+00
-9.15479645E-01  8.85833624E-01  0.00000000E+00
-1.03439821E-01  3.77243731E+00  0.00000000E+00
END
```

GRADIENT

GRADIENT is a list keyword that specifies the gradient components of the electronic structure theory Shepard point.

Example:

```
GRADIENT
 -2.80934625E-08 -7.33177670E-09  2.12772599E-11
 -1.41410517E-08  3.67943859E-08  1.46347707E-11
  4.22345150E-08 -2.94626079E-08 -3.59120307E-11
END
```

HESSIAN

HESSIAN is a list keyword that specifies the Hessian components of the electronic structure theory Shepard point.

Example:

```
HESSIAN
  7.26849015E-03 -2.46048400E-02  9.35414373E-02
 -4.14090454E-12  5.92075782E-12 -1.80332522E-08
 -1.58285758E-02  5.21279047E-02 -6.41965167E-12
  2.30461818E-02 -6.01027280E-03  1.56393041E-02
 -4.61939022E-11 -2.75845692E-02 -1.39825157E-01
  1.52073611E-11 -3.72014964E-11 -2.99670799E-08
  1.54332795E-11  1.38812605E-10  8.11309802E-08
  8.56008545E-03 -2.75230647E-02  1.05605569E-11
 -7.21760599E-03  3.35948420E-02 -3.06406234E-11
 -1.34247944E-03  3.06151128E-02 -1.09180741E-01
  4.02730797E-11 -2.45433355E-02  1.24185853E-01
 -1.01611061E-10 -6.07177726E-03 -1.50051120E-02
 -1.10664188E-11  3.12807014E-11  4.74362771E-08
 -9.01366720E-12 -9.26186406E-11 -5.11638840E-08
  2.00800682E-11  6.13379559E-11  3.77438791E-09
END
```

ICSHZERO

ICSHZERO is a list keyword that specifies the internal coordinates whose quadratic Taylor coefficients are set to zero, namely $C^{(k)}_{\alpha\beta} = 0$ ($\beta = 1, 2, \cdots, \Gamma$), where α labels the one of the internal coordinates specified by ICSHZERO (See Sec. V). In the EE-MCSI calculations, the quadratic couplings between the internal coordinates and electrostatic potentials are also set equal to zero. The format is the same as the case of ICSHEPARD keyword in the `esp.fu85` input file.

Example:

```
ICSHZERO
 5-8
 7-9 7-10 7-12
 2-1-5-6 3-1-5-6 4-1-5-6
END
```

ICSAZERO

ICSAZERO is a list keyword that specifies the internal coordinates whose linear and quadratic Taylor coefficients are set to zero, namely $b^{(k)}_{\alpha} = 0$ and $C^{(k)}_{\alpha\beta} = 0$ ($\beta = 1, 2, \cdots, \Gamma$), where α is a label for one of the internal coordinates specified by ICSAZERO (See Sec. V). In the EE-MCSI calculations, the quadratic couplings between the internal coordinate and electrostatic potentials are also set equal to zero. The format is the same as the case of ICSHEPARD keyword in the `esp.fu85` input file.

Example:

```
ICSAZERO
 5-8
 7-9 7-10 7-12
 2-1-5-6 3-1-5-6 4-1-5-6
END
```

QPHI (required for an EE-MCSI calculation)

QPHI is a list keyword that specifies the partial charges and electrostatic potentials of the electronic-structure-theory Shepard point. The format and units for the list are specified by FORMQPHI and UNITELP in the EEGEN83 section.

Example:

```
QPHI
 1 -0.2600038533E-01  0.0000000000E+00
 2  0.1185141676E+00  0.0000000000E+00
 3  0.1185141676E+00  0.0000000000E+00
 4  0.1185141676E+00  0.0000000000E+00
 5 -0.6647710587E+00  0.0000000000E+00
 6 -0.6647710587E+00  0.0000000000E+00
END
```

DQDR (required for an EE-MCSI calculation)

DQDR is a list keyword that specifies the component of the derivatives of charges with respect to coordinates, $\kappa_{ab} \equiv \dfrac{\partial Q_a}{\partial R_b} = \dfrac{\partial^2 V^{\mathrm{EEQM}}}{\partial \Phi_a \partial R_b}$, of the electronic-structure-theory Shepard point. The format and units for the list are specified by FORMDQDR and UNITDQDR in the EEGEN83 section.

Example:

```
DQDR
 1  1 0.00000000E+00 0.00000000E+00 0.00000000E+00 1.71480368E-02-2.97012710E-02
 1  2 0.00000000E+00 1.71480368E-02 2.97012710E-02 0.00000000E+00-3.42960737E-02
 1  3 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 9.77987307E-02
 1  4 0.00000000E+00 0.00000000E+00-9.77987307E-02
 2  1 1.98462395E-02-3.43746951E-02 0.00000000E+00-3.52215916E-02 6.10055862E-02
 2  2 0.00000000E+00 1.63732791E-02-1.21562599E-02 0.00000000E+00 2.34099030E-03
 2  3-2.02578056E-02 0.00000000E+00-1.44985822E-03 2.51122810E-03 1.30910625E-02
 2  4-1.44985822E-03 2.51122810E-03-1.30910625E-02
 3  1 1.98462395E-02 3.43746951E-02 0.00000000E+00 1.63732791E-02 1.21562599E-02
 3  2 0.00000000E+00-3.52215916E-02-6.10055862E-02 0.00000000E+00 2.34099030E-03
 3  3 2.02578056E-02 0.00000000E+00-1.44985822E-03-2.51122810E-03 1.30910625E-02
 3  4-1.44985822E-03-2.51122810E-03-1.30910625E-02
 4  1-3.96924789E-02 0.00000000E+00 0.00000000E+00-1.87142694E-02-8.10154572E-03
 4  2 0.00000000E+00-1.87142694E-02 8.10154572E-03 0.00000000E+00 7.04431833E-02
 4  3 0.00000000E+00 0.00000000E+00 2.89971644E-03 0.00000000E+00 1.30910625E-02
 4  4 2.89971644E-03 0.00000000E+00-1.30910625E-02
 5  1 0.00000000E+00 0.00000000E+00 3.87749919E-01 1.02072717E-02-1.76795132E-02
 5  2-1.81757312E-02 1.02072717E-02 1.76795132E-02-1.81757312E-02-2.04145434E-02
 5  3 0.00000000E+00-1.81757312E-02 0.00000000E+00 0.00000000E+00-2.35147321E-01
 5  4 0.00000000E+00 0.00000000E+00-9.80754037E-02
 6  1 0.00000000E+00 0.00000000E+00-3.87749919E-01 1.02072717E-02-1.76795132E-02
```

```
 6  2 1.81757312E-02 1.02072717E-02 1.76795132E-02 1.81757312E-02-2.04145434E-02
 6  3 0.00000000E+00 1.81757312E-02 0.00000000E+00 0.00000000E+00 9.80754037E-02
 6  4 0.00000000E+00 0.00000000E+00 2.35147321E-01
END
```

DQDPHI (required for an EE-MCSI calculation)

DQDPHI is a list keyword that specifies the component of the derivatives of charges with respect to electrostatic potentials, $\chi_{ab} \equiv \frac{\partial Q_a}{\partial \Phi_b} = \frac{\partial^2 V^{\mathrm{EEQM}}}{\partial \Phi_a \partial \Phi_b}$, of the electronic-structure-theory Shepard point. The format and units for the list are specified by FORMDQDPHI and UNITDQDPHI in the EEGEN83 section.

Example:

```
DQDPHI
 1  1-3.19481334E+00 6.53679089E-01 6.53679089E-01 6.53679089E-01 6.16888035E-01
 1  2 6.16888035E-01
 2  1 6.53679089E-01-9.08847834E-01 2.88245436E-02 2.88245436E-02 9.87598291E-02
 2  2 9.87598291E-02
 3  1 6.53679089E-01 2.88245436E-02-9.08847834E-01 2.88245436E-02 9.87598291E-02
 3  2 9.87598291E-02
 4  1 6.53679089E-01 2.88245436E-02 2.88245436E-02-9.08847834E-01 9.87598291E-02
 4  2 9.87598291E-02
 5  1 6.16888035E-01 9.87598291E-02 9.87598291E-02 9.87598291E-02-1.20630830E+00
 5  2 2.93140774E-01
 6  1 6.16888035E-01 9.87598291E-02 9.87598291E-02 9.87598291E-02 2.93140774E-01
 6  2-1.20630830E+00
END
```

D. Description of `esp.fu84` input file

The `esp.fu84` input file is a formatted input file that includes the geometry information for the points where MCSI or SCMM calculations should be performed. All the records are read in free format. The input has the same format for both symmetrized and non-symmetrized calculations.

The first two lines of the `esp.fu84` input file should contain:

- First line: The keyword POINTS followed by mcnpts, which is an integer indicating the number of points for which MCSI or SCMM are desired.
- Second line: The keyword FORMAT followed by mcifrm, which is an integer indicating the format used to read the data in the esp.fu84 input file.

Example:

```
POINTS 25
FORMAT 2
```

At this time there is a loop that goes from 1 to `mcnpts`, reading the information for each of the points. Therefore, there must be `mcnpts` blocks. All the information following these blocks is ignored.

For `mcifrm = 1`, each of these `mcnpts` blocks contains the following records:

- Record 1: `mcnatm`, an integer indicating the number of atoms in the system.
- Record 2: Comment line.
- Record 3: `mcnatm` lines indicating the index of the atom in the molecule (or the chemical symbol) and the Cartesian coordinates for that atom.

For `mcifrm = 2`, each of these `mcnpts` blocks contains the following records:

- Record 1: Comment line.
- Record 2: `mcnatm` lines indicating the index of the atom in the molecule (or the chemical symbol) and the Cartesian coordinates for that atom.

For `mcifrm = 3`, each of these mcnpts blocks contains the following records:

Record 1: Comment line. Record 2: `mcnatm` lines indicating the Cartesian coordinates for that atom.

For each of the formats (`mcifrm = 1, 2, or 3`) these records have to be repeated for each point until the information for all `mcnpts` points is input. Note that everything that is written on a line following the third Cartesian coordinate is ignored in a MM or MCSI calculation. (For case of an EE-MM or EE-MCSI calculation, see below.)

Example of format 1:

```
points 2
format 1
   7
Comment line: first geometry
1   0.000000  0.000000  0.000000
2   1.105391  0.000000  0.000000
3   1.452921  1.160817  0.000000
4   1.536666 -0.459271 -0.908287
5   1.540129 -0.398133  0.937112
6   1.386584  2.344293 -0.681525
7   0.850429  2.088654 -1.487727
   7
Comment line: second geometry
H   0.000000  0.000000  0.000000
C   1.205391  0.000000  0.000000
H   1.452921  1.160817  0.000000
H   1.536666 -0.459271 -0.908287
H   1.540129 -0.398133  0.937112
O   1.386584  2.344293 -0.681525
H   0.850429  2.088654 -1.487727
```

Example of format 2:

```
Points 2
Format 2
Comment line: first geometry
1   0.000000  0.000000  0.000000
2   1.105391  0.000000  0.000000
3   1.452921  1.160817  0.000000
4   1.536666 -0.459271 -0.908287
5   1.540129 -0.398133  0.937112
6   1.386584  2.344293 -0.681525
7   0.850429  2.088654 -1.487727
Comment line: second geometry
H   0.000000  0.000000  0.000000
C   1.205391  0.000000  0.000000
H   1.452921  1.160817  0.000000
H   1.536666 -0.459271 -0.908287
H   1.540129 -0.398133  0.937112
O   1.386584  2.344293 -0.681525
H   0.850429  2.088654 -1.487727
```

Example of format 3:

```
POINTS 2
FORMAT 3
Comment line: first geometry
 0.000000  0.000000  0.000000
 1.105391  0.000000  0.000000
 1.452921  1.160817  0.000000
 1.536666 -0.459271 -0.908287
 1.540129 -0.398133  0.937112
 1.386584  2.344293 -0.681525
 0.850429  2.088654 -1.487727
Comment line: second geometry
 0.000000  0.000000  0.000000
 1.205391  0.000000  0.000000
 1.452921  1.160817  0.000000
 1.536666 -0.459271 -0.908287
 1.540129 -0.398133  0.937112
 1.386584  2.344293 -0.681525
 0.850429  2.088654 -1.487727
```

In an EE-MM or EE-MCSI calculation (that is, when EECALC in the EEGENERAL section in the `esp.fu85` input file is true), the number after the Cartesian coordinate is read as the electrostatic potential at the center (nucleus) of that atom. The unit of the electrostatic potential is specified by UNITELPIN in the EEGENERAL section in the `esp.fu85` input file. If there is no number after the Cartesian coordinate, the electrostatic potential at the center of that atom is regarded as zero.

Example:

```
POINTS 2
FORMAT 2
*  transition state in gas phase
C     0.00000000   0.00000000   0.00000000
H    -0.53435591   0.92553159   0.00000000
H    -0.53435591  -0.92553159   0.00000000
H     1.06871182   0.00000000   0.00000000
CL    0.00000000   0.00000000   2.31503001
CL2   0.00000000   0.00000000  -2.31503001
*  transtion state in solution
C     0.00000000   0.00000000   0.00000000   4.75122986
H    -0.53520866   0.92700860   0.00000000   4.60761341
H    -0.53520866  -0.92700860   0.00000000   4.60761341
H     1.07041733   0.00000000   0.00000000   4.60761341
CL1   0.00000000   0.00000000   2.31589993   5.21148000
CL2   0.00000000   0.00000000  -2.31589993   5.21148000
```

E. Description of `esp.fu85` input file

The input for FORTRAN unit `esp.fu85` is in the free

to four sections. These are defined in the three subsections that follow this section. The input file consists of section

headers (section headers are always preceded by a star, *) and associated keywords.

The keywords are of two types: variable and list. A variable keyword always requires an argument. The argument must appear on the same line as the variable keyword. A list keyword has the form:

```
LISTKEYWORD
...
...
END
```

where the END line is mandatory.

All variable keywords have default values, and these types of keywords need to be given only if the user wishes to override the default. List keywords may have defaults as well, but in the current version of the code, only the MCTITLE keyword has default. List keywords start a subsection that is always terminated with an END line.

Except for the input of the MCTITLE keyword in the MCGENERAL section, all input is case insensitive, and anything after a pound sign (#) on a line is assumed to be a comment (except within the ICSHEPARD and ICDISTANCE list keywords). Blank lines are ignored. It is highly recommended that the novice user print out one of the test run input files and compare it with the descriptions in the following subsections. The rest of this section details the keywords for each of the three sections. Please note that every keyword must be used in its proper section, i.e., between the occurrence of its starred header and the occurrence of the next starred header (or, for the last section, between the proper starred header and the end of the file).

1. The MCGENERAL section

The MCGENERAL section is used to give a title that can be used to identify the run, to define all the atoms in the system, and to define the type of calculation that is to be carried out. The table below lists all valid keywords for this section. A more complete explanation for each keyword is given in the alphabetical glossary that follows.

Keyword	Type	Default	Description
MCATOMS	list	required	atoms in system
MCTITLE	list	blank	title with 5 lines maximum
ORDERCALC	variable	*mcmm*	carry out an MCSI calculation
SCMASS	variable	*1.0*	scaling factor for mass-scaled coordinates
SSWEIGHT	variable	*0.000001*	step size (in a.u.) for numerical calculations of the first and second derivatives of the weight function with respect to internal coordinates
STEPSIZE	variable	*0.000001*	step size (in a.u.) for numerical calculations of MCSI gradients and Hessians
TYPECALC	variable	*energy*	calculate of energy only
UDELTA	variable	2.56×10^{-8}	a value for Δ^2 (in a.u.) in eqs 15 and 52 of the Appendix
UNITINPUT	variable	*ang*	specify the units for the geometries given in the `esp.fu84` file
UNITOUTPUT	variable	*bohr*	specify the units for the results printed in the `esp.fu86` file
UNN	variable	*2.0*	a value for n in eqs 15 and 52 of the Appendix

Glossary of MCGENERAL keywords

MCATOMS

MCATOMS is a required list keyword that is used to specify all the atoms in the system. For each atom the user must specify n, a sequential number unique to each atom in the molecule, and a label, either the atomic number or symbol of atom n. The mass of the atom in atomic mass units (amu) can also be specified following the label. If it is not included then the mass of the most prominent isotope is used. This program uses the universal atomic mass scale, i.e., the mass of 12 C is 12.000000. (For this scale, the correct abbreviation is u, although most people use the old-fashioned abbreviation amu.) The mass is supplied from an internally stored table using the atomic number or symbol as given by the user. Note that the unique number n will be used in other sections to specify the atom. The example below corresponds to monodeuterated methane.

Example:

```
MCATOMS
1 C
2 H
3 H
4 H 1.0078
5 H 2.0140
END
```

Note: The ordering of atoms defined in the MCGENERAL section has to be the same as in the `esp.fu81` through `esp.fu84` input files.

MCTITLE

MCTITLE is a list keyword that allows the user to give a title to the run. The user can specify up to five lines of 80 characters each. Like all list variables, an END must be given at the end of the MCTITLE subsection.

Example:

```
MCTITLE
 HO - H - CH3 System
 Resonance function constant at 28 kcal/mol
END
```

ORDERCALC

ORDERCALC is a variable keyword that specifies whether an SCMM or an MCSI calculation is to be carried out. The default is `mcmm`.

Option	Description
mcmm	MCSI calculation
scmm1	SCMM calculation with configuration 1 bonding scheme
scmm2	SCMM calculation with configuration 2 bonding scheme

Example:

```
ORDERCALC scmm2
```

SCMASS

SCMASS is a variable keyword that specifies the value (in amu) for the parameter μ used in defining the mass-scaled coordinates. (Mass-scaled coordinates, $\mathbf{x}$, are defined as $x_i = (\frac{m_i}{\mu})^{1/2} R_i$, where m_i is the mass of atom $i = 1 \cdots N$, and R is one Cartesian coordinate.) The default is `1.0` amu.

Example:

```
SCMASS 12.0
```

SSWEIGHT

This variable keyword specifies the step size which is used in numerical evaluation of the first and second derivatives of the weight function with respect to the internal coordinates. Note that in general an MCSI gradient and Hessian is calculated semi-analytically as described in Secs. III and IV. This step size is only used in a single numerical step to obtain $\frac{\partial w_k}{\partial \mathbf{r}}$ and $\frac{\partial^2 w_k}{\partial \mathbf{r}^2}$.

STEPSIZE

STEPSIZE specifies the step size for numerical calculations of the first of second derivatives of the MCSI energy; this keyword is used only if the TYPECALC keyword is set to a `gradtest` or to a `hesstest`. The default value for a stepsize is `0.000001` a.u.

TYPECALC

TYPECALC is a variable keyword that specifies whether an energy, gradient, or Hessian is to be carried out. One can also calculate the first ant second derivatives of the MCSI or EE-MCSI energy both analytically and numerically for comparison. The default is `energy`.

Option	Description
energy	energy calculation
gradient	energy and gradient calculation
hessian	energy, gradient, and Hessian calculation
gradtest	energy, analytical and numerical gradient
hesstest	energy, analytical and numerical Hessian
chargetest	energy, analytical and numerical charge
dqdrtest	energy, analytical and numerical CRK $\boldsymbol{\kappa}$
dqdphitest	energy, analytical and numerical CRK $\boldsymbol{\chi}$

Example:

```
TYPECALC Hessian
```

UDELTA and UNN

These variable keywords specify the values for Δ^2 and n, respectively, that that are used in eqs. 51 and 97. We note that some experimenting may be required to find the best values for these parameters for a particular reaction.

UNITINPUT

UNITINPUT is a variable keyword that specifies the units for the geometries given in the `esp.fu84` input file. (This keyword has no effect on the units for the geometries given in the `esp.fu81` and `esp.fu82` input files, which should be in angstroms, and the `esp.fu83` input file, which are defined in the MCGEN83 section.) The default is `ang`.

Option	Description
bohr	coordinates are in unscaled bohrs
msbohr	coordinates are in mass-scaled bohrs
ang	coordinates are in unscaled angstroms
msang	coordinates are in mass-scaled angstroms

Example:

```
UNITINPUT bohr
```

UNITOUTPUT

UNITOUTPUT is a variable keyword that specifies the units for the results printed in the `esp.fu86` output file. The `esp.fu86` output file gives the coordinates in angstroms and bohrs, and the energies in hartrees, eV, cm^1, and kcal/mol for any type of calculation. For gradient and Hessian calculations, the results (geometry, gradient, and Hessian) are given in a format similar to the one used in a Gaussian formatted checkpoint file. The UNITOUTPUT keyword applies to these results. The default is `bohr`.

Option	Description
bohr	coordinates are in unscaled bohrs, gradients are in hartrees per unscaled bohr, Hessians are in hartrees per unscaled bohr^2
msbohr	coordinates are in mass-scaled bohrs, gradients are in hartrees per mass-scaled bohr, Hessians are in hartrees per mass-scaled bohr^2
ang	coordinates are in unscaled angstroms, gradients are in hartrees per unscaled angstroms, Hessians are in hartrees per unscaled angstroms^2
msang	coordinates are in mass-scaled angstroms, gradients are in hartrees per mass-scaled angstroms, Hessians are in hartrees per mass-scaled angstroms^2

Example:

```
UNITOUTPUT msbohr
```

2. The MCENERGETICS section

The MCENERGETICS section is used to specify single configuration energies at the energy reference states/points and the electronic structure theory energy difference between these states/points. The table below lists all valid keywords for this section. A more complete explanation for each keyword is given in the alphabetical glossary that follows.

Keyword	Type	Default	Description
ZERO1	variable	*0.0*	molecular mechanics energy of the configuration 1 energy reference state/point (in hartrees)
ZERO2	variable	*0.0*	molecular mechanics energy of the configuration 2 energy reference state/point (in hartrees)
EDIFF	variable	*0.0*	electronic structure theory energy difference (in hartrees)

Glossary of MCENERGETICS keywords

EDIFF

EDIFF is a variable keyword whose argument specifies the electronic structure energy difference (in hartrees) between the configuration 2 energy reference state/point and the configuration 1 energy reference state/point. The default is `0.0`.

Example:

```
EDIFF 0.0125479
```

ZERO1

ZERO1 is a variable keyword whose argument specifies the molecular mechanics energy of the configuration 1 energy reference state/point (in hartrees). The default is `0.0`.

Example:

```
ZERO1 0.0035921
```

ZERO2

ZERO2 is a variable keyword whose argument specifies the molecular mechanics energy of the configuration 2 energy reference state/point (in hartrees). The default is `0.0`.

Example:

```
ZERO2 0.0012528
```

3. The SYMMETRY section

The SYMMETRY section is used to specify some parameters required for symmetrized calculations. This section

should be omitted in nonsymmetrized calculations. The table below lists all valid keywords for this section.

Keyword	Type	Default	Description
ALPHA1	variable	*0.0*	A parameter for $\delta = \frac{1}{\alpha}$ used in eqs. (10-12) and in eqs. (31-33) of Ref. 4 for MM configuration 1 (in kcal/mol)
ALPHA2	variable	*0.0*	Same as ALPHA1 but for MM configuration 2
SIGMA1	variable	*1*	Symmetry factor for MM configu-ration 1, i.e., the number of times the lowest-energy configuration 1 occurs among the $m!$ permutations of the labels of identical atoms. This is explained in Ref. 4.
SIGMA2	variable	*1*	Same as SIGMA1 but for MM configuration 2
IPERMSYM	variable	*0*	Indicates whether symmetrized (1) or nonsymmetrized (0) calculations will be performed.

4. The RESONANCE section

The RESONANCE section is used to specify the method used for calculating the resonance energy function (also called the resonance integral or the diabatic coupling), V_{12}. This section is not needed if the ORDERCALC keyword in MCGENERAL section is set to `scmm1` or `scmm2` (as in a single-configuration molecular mechanics calculation using configuration 1 or 2, respectively). The table below lists all valid keywords for this section. A more complete explanation for each keyword is given in the alphabetical glossary that follows:

Keyword	Type	Default	Description
ICDISTANCE	list	not used	define internal coordinates used in calculating generalized distances
ICSHEPARD	list	not used	define internal coordinates used in Shepard interpolation
ISHMM	variable	*true*	the coordinates given in the `esp.fu81` and `esp.fu82` are used as SCMM Shepard points, that is, as points where V_{12} and its Taylor series are taken to be identically zero
JACOBIANS	variable	*0*	use (=*1*) or not to use (=*0*) Jacobians and Hessians for an atom transfer reaction (see Sec. II B for more details)
LINEARITY	switch	*nonlinear*	system is non-linear
NONHERMITIAN	switch	*true*	use non-Hermitian formalism
POINTEXP	variable	*1*	the electronic structure theory point from which the expansion will be used to determine V_{12}
PRINTIC	variable	none	do not print internal coordinates
RESFORM	variable	*2*	quadratic form used for V_{12}
RESMETHOD	variable	*shepard*	the method used for determining V_{12}
RESVALUE	variable	*0.03187203*	the constant value (in hartrees) of V_{12}

Glossary of RESONANCE keywords

ICDISTANCE

ICDISTANCE is a list keyword used to specify the curvilinear internal coordinates used for estimating the generalized distances between points for the Shepard interpolation (set **s**). Similar to the ICSHEPARD keyword, the allowed types are distances $(i - j)$, angles $(i - j - k)$, dihedral angles $(i - j - k - l)$, sines of dihedral angles $(m - i - j - k - l)$ and degenerate angles $(i = j = k)$ where i, j, k, and l are the atom-number-

labels defined in the MCGENERAL section, and m is periodicity of the dihedral angle (See Sec V). Note: when the bond angle is close to 180 degrees (i.e., greater than about 175 degrees), the use of a doubly degenerate linear bending coordinate is preferred to a "normal" bending coordinate. Note also that sines of dihedral angles are available for nonsymmetrized calculations. There are no defaults for this keyword, and this keyword is required if RESMETHOD is set to expansion or shepard.

Example:

```
ICDISTANCE
2-5 2-6 6-5
END
```

Note: The pound sign (#) cannot be used as a comment card within ICDISTANCE keyword list.

ICSHEPARD

ICSHEPARD is a list keyword used to specify the curvilinear internal coordinates used for the transformation step and the Shepard interpolation in internal coordinates (set **r**). Similar to the ICDISTANCE keyword, the allowed types are distances ($i-j$), angles ($i-j-k$), dihedral angles ($i-j-k-l$), sines of dihedral angles ($m-i-j-k-l$), and degenerate angles ($i=j=k$) where i, j, k, and l are the atom-number-labels defined in the MCGENERAL section, and m is periodicity of the dihedral angle (See Sec V). Note: when the bond angle is close to 180 degrees (i.e., greater than about 175 degrees), the use of a doubly degenerate linear bending coordinate is preferred to a "normal" bending coordinate. Note also that sines of dihedral angles are available for nonsymmetrized calculations, and that dihedral angles ($i-j-k-l$) and sines of dihedral angles ($m-i-j-k-l$) are allowed to be used at the same time in the current version of MCSI. There are no defaults for this keyword, and this keyword is required if RESMETHOD is set to expansion or shepard.

Example:

```
ICSHEPARD
2-1 3-2 6-2 3-6 2-4 5-2 7-6
1-2-3 3-2-4 1-2-4 2=4=5
1-2-4-5
END
```

Note: The pound sign (#) cannot be used as a comment card within ICSHEPARD keyword list.
Note: The current version of the code only supports cases when the coordinates given in the ICDISTANCE and ICSHEPARD cards are the same.

ISHMM

The ISHMM keyword is used to specify whether the coordinates given in the `esp.fu81` and `esp.fu82` files are used as SCMM Shepard points, where (and the Taylor series of is zero), or not. This option is available only for a nonsymmetrized calculation. Note that when ISHMM is true, the number of Shepard points is equal to the number of electronic-structure-theory Shepard points plus 2, whereas when ISHMM is `false`, the number of Shepard points is the same as the number of electronic-structure-theory Shepard points. The default is `true`.

Option	Description
true	the coordinates given in the `esp.fu81` and `esp.fu82` files are used as SCMM Shepard points
false	the coordinates given in the `esp.fu81` and `esp.fu82` files are not used as SCMM Shepard points

Example:

```
ISHMM false
```

JACOBIANS

The JACOBIANS keyword is used to specify whether or not to use the Jacobians and Hessians in calculation of the derivatives of the weight function with respect to internal coordinates. The reason why one needs Jacobians and Hessians is because Shepard interpolation is carried out in coordinates **r**, whereas the weight function is calculated in coordinates **s**. If **s** is different from **r** and is not a subset of **r**, one generally needs Jacobians $\frac{\partial s}{\partial r}$ and Hessians $\frac{\partial^2 s}{\partial r^2}$, if calculation of the gradient or Hessian of V is requested. For **s** being different from **r** (and not representing a subset of **r**), the present implementation of the MCSI code only supports the case suitable for an atom transfer reaction, as described in Sec. II B. If **s** (specified by ICDISTANCE) is the same as **r** (specified by ICSHEPARD) or if **s** is a subset of **r**, the JACOBIANS switch should be set to *0*. If one choses **s** and **r** as described in Sec. II B 1, the JACOBIANS switch should be set to 1. Internal coordinates **s** and **r** can be given in any order. Note: it is recommended to run GRADTEST and/or HESSTEST calculation prior to running actual calculations to make sure that the combination of keywords is appropriate.

Option	Description
0	do not use the Jacobians and/or Hessians
1	use the Jacobians and/or Hessians as described in Sec. II B 1

Example:

```
JACOBIANS         0
```

LINEARITY

The LINEARITY keyword is used to specify whether the system is linear or not. The default is `nonlinear`.

Option	Description
linear	the system is linear
nonlinear	the system is non-linear

Example:

```
LINEARITY linear
```

NONHERMITIAN

The NONHERMITIAN keyword is used to specify whether to use the non-Hermitian MCSI technique. The default is `true`.

Option	Description
true	use non-Hermitian MCSI technique (Ref. 5)
false	use Hermitian MCSI technique (Refs. 3 and 7)

Example:

```
NONHERMITIAN   true
```

POINTEXP

POINTEXP is a variable keyword that indicates the point in the `esp.fu83` input file that will be used to determine the resonance energy function by using the expansion around this point. Note that the entry for POINTEXP keyword should be smaller than the number of points defined in the `esp.fu83` input file (NESTSP). This keyword is ignored if RESMETHOD is not set to expansion. The default is `1`.

Example:

```
POINTEXP            2
```

PRINTIC

PRINTIC is a variable keyword that indicates if the internal coordinates used for Shepard interpolation and the internal coordinates used for generalized distances should be printed in the `esp.fu86` output file. The default is none.

Option	Description
none	do no print any internal coordinates
point	print the internal coordinates only for each point where an MCSI calculation is desired
all	print the internal coordinates for the Shepard points and for each point where an MCSI calculation is desired

Example:

```
PRINTIC   point
```

RESFORM

RESFORM is a variable keyword that indicates the form used for the resonance energy function. The default and the only available option in the current version of the code is `2`.

Option	Description
1	generalized Gaussian form
2	quadratic form

Example:

```
RESFORM   2
```

RESMETHOD

RESMETHOD is a variable keyword that indicates the method used for the determining the resonance energy function. The default is `shepard`.

Option	Description
constant	V_{12} has a constant value.
expansion	V_{12} is determined from an expansion around an electronic structure theory point. (This option is not currently supported.)
shepard	V_{12} is determined through Shepard interpolation.

Example:

```
RESMETHOD  expansion
```

RESVALUE

RESVALUE is a variable keyword whose argument specifies the constant value of the resonance energy function (in hartrees). This keyword is ignored if RESMETHOD is not set to constant. The default is `0.0318720274` (which is equal to 20 kcal/mol).

Example:

```
RESVALUE 0.0835691
```

5. The EEGENERAL section

The EEGENERAL section is used to specify whether an EE-MM or EE-MCSI calculation is carried out and to define some parameters for the EE-MM or EE-MCSI calculation. The table below lists all valid keywords for this section. A more complete explanation for each keyword is given in the alphabetical glossary that follows.

Keyword	Type	Default	Description
EECALC	variable	*false*	carry out the original MCSI
EETYP	variable	*charge*	calculate partial charges only
UNITELPIN	variable	*volt*	specify the units for the electrostatic potentials given in the `esp.fu84` file
ICEEZERO	list	none	define internal coordinates whose coefficients are set equal to zero in EE-MM or EE-MCSI diagonal elements calculations

Glossary of EEGENERAL keywords

EECALC

EECALC is a variable keyword that specifies whether an EE-MM or EE-MCSI calculation is to be carried out or not. In the current MCSI, only nonsymmetrized calculations are available for the EE-MCSI calculation. If ORDERCALC in the MCGENERAL section is `scmm1` or `scmm2`, an EE-MM calculation (based on the diagonal element of $\mathbf{V}^{\text{EE-MCSI}}$, that is, eq. 18) is to be carried out. The default is `false`.

Option	Description
true	EE-MM or EE-MCSI calculation
false	MM or original MCSI calculation

Example:

```
EECALC true
```

EETYPE

EETYPE is a variable keyword that specifies whether partial charges, $\mathbf{Q} = \dfrac{\partial V^{\text{EE-MCSI}}}{\partial \mathbf{\Phi}}$, derivatives of the charges with respect to coordinates, $\boldsymbol{\kappa} = \dfrac{\partial \mathbf{Q}}{\partial \mathbf{R}} = \dfrac{\partial^2 V^{\text{EE-MCSI}}}{\partial \mathbf{R} \partial \mathbf{\Phi}}$, or derivatives of the charges with respect to electrostatic potentials, $\boldsymbol{\chi} = \dfrac{\partial \mathbf{Q}}{\partial \mathbf{\Phi}} = \dfrac{\partial^2 V^{\text{EE-MCSI}}}{\partial \mathbf{\Phi} \partial \mathbf{\Phi}}$, are to be calculated in an EE-MM or EE-MCSI calculation. When $\boldsymbol{\kappa}$ is to be calculated, TYPECALC must be `gradient` or `hessian`. The default is `charge`.

Option	Description
charge	partial charges are to be calculated.
dqdr	partial charges and derivatives of partial charges with respect to coordinates $\boldsymbol{\kappa}$ are to be calculated.
dqdphi	partial charges and derivatives of partial charges with respect to electrostatic potentials $\boldsymbol{\chi}$ are to be calculated.
all	partial charges, derivatives of partial charges with respect to coordinates $\boldsymbol{\kappa}$, and derivatives of partial charges with respect to electrostatic potentials $\boldsymbol{\chi}$ are to be calculated.

Example:

```
EETYPE all
```

UNITELPIN

UNITELPIN is a variable keyword that specifies the units for the electrostatic potentials given in the `esp.fu84` input file. (This keyword has no effect on the units for the electrostatic potentials given in the `esp.fu81` and `esp.fu82` input files, which are defined in the CRKFMT section, or in the `esp.fu83` input file, which are defined in the EEGEN83 section.) The default is `volt`.

Option	Description
au	electrostatic potentials are in a.u.
volt	electrostatic potentials are in volts.

Example:

```
UNITELPIN volt
```

ICEEZERO

ICEEZERO is a list keyword that specifies the internal coordinates whose quadratic Taylor coefficients in the EE-MM or EE-MCSI diagonal calculations are set to zero, namely $(\boldsymbol{\kappa}_0)_{\alpha\beta} = 0 \; ; (\alpha = 1, 2, \cdots, N^{QM})$ in eqs. 19 and 27, where β is an internal coordinate specified by ICEEZERO. The format is the same as the case of ICSHEPARD keyword in the `esp.fu85` input file.

Example:

```
ICEEZERO
 2-1-5-6 3-1-5-6 4-1-5-6
 2-1-5-7 3-1-5-7 4-1-5-7
END
```

F. Description of `param.prm` input file

The parameter file for tinker, which includes all the force field parameters, is read from a file called `param.prm`. The format of this file is the same as in any TINKER calculation, with the only exception being the changes in the input for bond parameters in order to take advantage of the possibility of treating selected modes as Morse oscillators, as indicated above (see Sec. VIII A 1).

In particular, we note that a charmm27 force field parameter file `charmm27t35.prm` was generated for TINKER 3.5mn4 on the basis of the charmm27 parameter file `charmm27.prm` associated with TINKER 4.1. The atom types implemented in both `charmm27t35.prm` and `charmm27.prm` parameter files are the same. The implementation of force field parameters for TINKER 3.5mn4 is discussed in more detail in Sec. X.

In most cases, the user can use the parameter files provided with TINKER 3.5mn4 without any modification.

X. COMMENTS ON FORCE FIELDS

The force field parameter files in TINKER 3.5MN4 are generated on the basis of the force field parameter files distributed within TINKER 3.5, with only a few modifications. We have discussed in Sec. VIII A the modifications related to the MCSI calculations, which are concerned with (1) the Morse treatment for the stretching terms, (2) the MM3 van der Waals terms, and (3) the VESCF treatment for π-systems. These changes normally require revisions not only of parameter files but also of subroutines.

In this section, we discuss the implementation of force field parameters in TINKER 3.5MN4, with the emphasis on the differences between tinker and the original force field programs. In other words, the issues discussed here are concerned with the algorithms in TINKER rather than with the MCSI method.

For the most part the force fields are treated in TINKER in the same way as in the original programs. However, there are several exceptions, in particular for the CHARMM force field.

For the most part the force fields are treated in TINKER in the same way as in the original programs. However, there are several exceptions, in particular for the CHARMM force field.

- Completeness of force field

  For many force fields, TINKER does not implement the complete set of atoms types. Users are encouraged to check the tinker manual for more information.

- Atom classes

  TINKER uses atom class in additional to atom type to classify atoms. (See the TINKER 4.1 manual, page 45). Manipulation of atom types and the proliferation of parameters as atoms are further subdivided into new types is the major difficulty of force field calculation. For example, if each topologically distinct atom arising from the 20 natural amino acids is given a different atom type, then about 300 separate types are required (this ignores the different N- and C-terminal residues, diastereotopic hydrogens, etc.). However, these 300 types lead to thousands of different force field parameters. In fact, there are thousands of distinct torsional parameters alone. It is impossible at present to fully optimize each of these parameters; and even if that were done, a great many of the parameters would be nearly identical. Two approaches are available to handle the proliferation of parameters.

  The first is to specify the molecular fragments to which a given parameter can be applied in terms of a chemical structure language, smiles strings for example. Some commercial systems, such as the tripos Sybyl software, make use of such a scheme to parse structures and assign force field parameters.

  A second general approach is to use hierarchical cascades of parameter groups. tinker uses a simple version of this scheme. Each tinker force field atom has both an atom type number and an atom class number. The types are subsets of the atom classes, i.e., several different atom types can belong to the same atom class. Force field parameters that are somewhat less sensitive to local environment, such as local geometry terms, are then provided and assigned based on atom class. Other energy parameters, such as electrostatic parameters, those are very environment dependent are assigned over the atom types. This greatly reduces the number of independent multiple-atom parameters such as the four-atom torsional parameters.

  However, in TINKER 3.5 (and TINKER 3.5mn4 as well), atom class is not used. In other words, atom type is identical to atom class in `tinker` 3.5 (and tinker 3.5mn4). In later versions of `tinker` (e.g. TINKER 4.1), atom classes are employed. The current version of MCSI is based on TINKER 3.5mn4, and therefore makes use of atom types exclusively.

- Van der Waals (VDW) 1-4 terms

  In some force fields such as the CHARMM force field, a set of specially developed parameters is used instead of the normal VDW parameters to evaluate the VDW 1–4 interactions. However, the data structure in TINKER 3.5 does not provide entries for the VDW 1–4 parameters. There are no such VDW 1–4 parameters provided in those force field (e.g., CHARMM force field) parameter files, either. The VDW 1–4 interactions are not correctly calculated in TINKER 3.5 through TINKER 3.5mn3 when using those force fields.

  In TINKER 3.5mn4, the data structure was modified such that the use of the VDW 1–4 parameters is allowed. Some subroutines were also changed accordingly.

It should be noted that the later versions of tinker (e.g., tinker 4.1) do handle the VDW 1-4 interactions correctly. See Sec. VIII A 2 for special discussion of the MM3 van der Waals term.

- Improper dihedrals

  Improper dihedral angles (usually just called improper dihedrals) are treated different manner in different force fields. For example, in AMBER the improper terms have the same functional form as proper torsions, and there is no separate code for those in the energy evaluation routines, although the improper dihedrals need to be identified and given the correct parameters.

  CHARMM however uses a different functional form for improper dihedrals. Following CHARMM style, the improper dihedral for a trigonal atom D bonded to atoms A, B, and C could be input as improper dihedral angle D-A-B-C. The actual angle computed by the program is then literally the dihedral D-A-B-C, which will always have as its ideal value zero degrees. In general D-A-B-C is different from D-B-A-C; the order of the three peripheral atoms matters. (In the original charmm parameter files, the trigonal atom is often listed last; i.e., as C-B-A-D instead of D-A-B-C.)

  TINKER handles the charmm improper dihedrals in a different way from the original CHARMM. TINKER 3.5 (and TINKER 3.5mn4) uses only one improper parameter per site, the CHARMM force constants have been doubled in TINKER 3.5 (and TINKER 3.5mn4). According to Ponder, the differences between tinker and original charmm are in the hundredths of kcal/mol in the total energy.

  In TINKER, the energy of an improper dihedral is given by

$$E = k'_{DABC}(\phi_{DABC} - \phi_{DABC_0})^2, \tag{118}$$

  where $k'_{DABC} = 2k_{DABC}$, k_{DABC} is the CHARMM force constant for improper dihedral D-A-B-C (actually the C-B-A-D in CHARMM notation), ϕ_{DABC} and ϕ_{DABC_0} are the instantaneous and equilibrium values for the improper dihedral D-A-B-C.

  For the users information, we note that the treatments of charmm improper dihedrals in TINKER are slightly different in version 3.5 and later versions (e.g., version 4.1). Therefore, one does not necessarily obtain identical results when using different versions of `tinker` even if the same CHARMM force field (e.g., charmm27) is employed.

  The treatment outlined above, equation 118, is applied in both TINKER 3.5 and 4.1. The difference between TINKER 3.5 and 4.1 is that symmetry is further considered in TINKER 4.1. If the atom classes of atoms A and B are the same, energy will be calculated by

$$\begin{aligned} E = {} & k_{DABC}\left(\phi_{DABC} - \phi_{DABC_0}\right)^2 \\ & + k_{DABC}\left(\phi_{DBAC} - \phi_{DBAC_0}\right)^2. \end{aligned} \tag{119}$$

  Similarly, when atom classes of A and C (or B and C) are the same:

$$\begin{aligned} E = {} & k_{DABC}\left(\phi_{DABC} - \phi_{DABC_0}\right)^2 \\ & + k_{DABC}\left(\phi_{DCBA} - \phi_{DCBA_0}\right)^2. \end{aligned} \tag{120}$$

  or

$$\begin{aligned} E = {} & k_{DABC}\left(\phi_{DABC} - \phi_{DABC_0}\right)^2 \\ & + k_{DABC}\left(\phi_{DACB} - \phi_{DACB_0}\right)^2. \end{aligned} \tag{121}$$

  And if atom types of A, B, and C are the same, one has

$$\begin{aligned} E = {} & k''_{DABC}\left(\phi_{DABC} - \phi_{DABC0}\right)^2 \\ & + k''_{DABC}\left(\phi_{DACB} - \phi_{DACB0}\right)^2 \\ & + k''_{DABC}\left(\phi_{DBAC} - \phi_{DBAC0}\right)^2 \\ & + k''_{DABC}\left(\phi_{DBCA} - \phi_{DBCA0}\right)^2 \\ & + k''_{DABC}\left(\phi_{DCAB} - \phi_{DCAB0}\right)^2 \\ & + k''_{DABC}\left(\phi_{DCBA} - \phi_{DCBA0}\right)^2, \end{aligned} \tag{122}$$

  where $k''_{DABC} = k_{DABC}/3$.

  Please also note that the parameter k_{DABC} is identical in TINKER 3.5 and TINKER 4.1. However, neither treatment (in TINKER 3.5 or 4.1) is identical to the original CHARMM treatment. The treatment in TINKER 3.5mn4 is the same as that in TINKER 3.5.

XI. SAMPLE TEST RUNS

The test suite includes fifteen test runs. Each of these test runs is described below. To run the complete test suite, one can run either one of the scripts located in the `testmct` directory, `run_all` or `run_t_all` (the `run_t_all` script also does the timing of the runs).

To the convenience of the user, we provide two scripts, `getnew` and `compnew`, for doing comparisons. Running script `getnew` will collect the most recent test calculation outputs into directory `testonew`, and subsequently running script `compnew` will generate files for comparisons between these newly obtained and those distributed outputs for the test runs.

Note that the gradient and Hessian results are printed in the Gaussian formatted checkpoint file format. The very small values of some gradient and Hessian components are subject to machine-dependent round-off errors and are not expected to be exact compared to the values in the distributed output files that are obtained on an IBM Regatta Power4 machine.

All test runs in this version of MCSI use the default value

of the NONHERMITIAN keyword, that is, they use the formalism of Ref. 5.

A. Test run 1

Nonsymmetrized MCSI energy calculation on the Cl – H – Br system

This test run is an MCSI energy calculation (the default) for the Cl – H – Br system. The chemical process modeled in this calculation is Cl + H–Br → Cl–H + Br. The configuration 1 has the hydrogen atom bonded to bromine atom while the configuration 2 has the hydrogen atom bonded to chlorine. The configuration 1 energy reference state is the optimized H–Br and Cl atom infinitely separate, and the configuration 2 energy reference state is the optimized H–Cl and Br atom infinitely separate. The energy difference between these energy reference states became the energy of reaction. This is one of the systems treated in Ref. 28 We use the reaction energy of -9.18 kcal/mol, which is the calculated value at the MP2/6-31G(d) level of theory used here. V_{12} is determined using Shepard interpolation with eleven electronic structure theory Shepard points and two SCMM points (in this case they are the structures of the van der Waals wells). The expansions in Shepard interpolation use three internal coordinates (set **r**: Cl–H, and H–Br distances, and Cl–H–Br angle), which are different than the ones used in calculating the generalized distance (set s: Cl–H, H–Br, and Cl–Br distances) between points. Since **s** is different from **r**, the JACOBIANS keyword in the RESONANCE Section is set to 1. The geometry input file (`esp.fu84` or `test1.84`) contains three geometries (given in angstrom), and the MCSI calculation is carried out for each of these points. The value of the parameter D (see Sec. VIII A 2) for this test run is 0.01.

INPUT FILES:

`test1.81`	Configuration 1 input file
`test1.82`	Configuration 2 input file
`test1.83`	Electronic structure theory data file
`test1.84`	Geometry input file
`test1.85`	General input file
`param-test1.prm`	Modified tinker parameter file
`run_test1`	Script file to run mcsi
`run_t_test1`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test1.86`	Full output file

B. Test run 2

Nonsymmetrized MCSI gradient calculation on the O – H – CH_3 system

This test run is an MCSI gradient calculation for the O – H – CH_3 system. The chemical reaction modeled in this calculation is O(3P) + CH_4 → O–H + CH_3, in which the configuration 1 has the transferring hydrogen atom bonded to carbon while the configuration 2 has this hydrogen atom bonded to oxygen. This is one of the systems treated in Ref. 28 V_{12} is determined using Shepard interpolation with eleven electronic structure theory points and two SCMM points (these two points are the van der Waals wells in this case). The Shepard interpolation is carried out by using fourteen internal coordinates; these internal coordinates form set **r**. Two of these (O–Ht, and Ht–C distances, where Ht represents the transferring hydrogen) are included in set **s** and are used in calculating the generalized distance. The geometry input file (`esp.fu84` or `test2.84`) contains two geometries (given in angstrom), and the MCSI calculation is carried out for both of these points. The value of the parameter D (see Sec. VIII A 2) for this test run is 0.01.

INPUT FILES:

`test2.81`	Configuration 1 input file
`test2.82`	Configuration 2 input file
`test2.83`	Electronic structure theory data file
`test2.84`	Geometry input file
`test2.85`	General input file
`param-test2.prm`	Modified tinker parameter file
`run_test2`	Script file to run mcsi
`run_t_test2`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test2.86`	Full output file

C. Test run 3

Nonsymmetrized MCSI gradient calculation on the O – H – CH_3 system

This test run is an MCSI gradient calculation for the O – H – CH_3 system. This test run is the same as the test run 2 except it uses six Shepard points; these consist of four electronic structure theory Shepard points, and two single-configuration molecular mechanics Shepard points, namely SCMM point for configuration 1 and SCMM point for configuration 2. Note that the `test3.83` file has the same electronic structure information as the `test2.83` file (except the value for NESTSP) but only four electronic structure theory points are actually used in the calculation, the ones with indexes greater than NESTSP are ignored.

INPUT FILES:

`test3.81`	Configuration 1 input file
`test3.82`	Configuration 2 input file
`test3.83`	Electronic structure theory data file
`test3.84`	Geometry input file
`test3.85`	General input file
`param-test3.prm`	Modified tinker parameter file
`run_test3`	Script file to run mcsi
`run_t_test3`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test3.86`	Full output file

D. Test run 4

Nonsymmetrized MCSI Hessian calculation on the HO – H – C_3H_7 system

This test run is an MCSI Hessian calculation for the HO – H – C_3H_7 system. Configuration 1 has the transferring hydrogen atom bonded to the secondary carbon of propane while configuration 2 has this hydrogen atom bonded to oxygen. The chemical reaction modeled in this calculation is HO + C_3H_8 → H_2O + C_3H_7. The configuration 1 energy reference state is represented by HO and C3H8 in their equilibrium structure and infinitely separate, and the configuration 2 energy reference state is represented by H_2O and C_3H_7 in their equilibrium structure and infinitely separate. The energy difference between these energy reference states is the energy of reaction. This is one of the systems treated in Ref. 28 The electronic structure theory used is MPW1K/6-31+G(d,p), and the energy of reaction is determined to be -16.51 kcal/mol. V_{12} is determined using Shepard interpolation with eight points, six electronic structure theory points and two SCMM points. The Shepard interpolation is carried out by using thirty-eight internal coordinates (set **r**). Two of these (O–Ht, and Ht–C distances, where Ht represents the transferring hydrogen) are included in the set **s** that is used to calculate the generalized distance. Note also that the molecular mechanics energy of the optimized propane and the optimized sec-propyl a are not zero so ZERO1 and ZERO2 keywords are included in MCENERGETICS section of the `esp.fu85` input file to overwrite the default values. The value of the parameter D (see Sec. VIII A 2) for this test run is 0.01.

INPUT FILES:

`test4.81`	Configuration 1 input file
`test4.82`	Configuration 2 input file
`test4.83`	Electronic structure theory data file
`test4.84`	Geometry input file
`test4.85`	General input file
`param-test4.prm`	Modified tinker parameter file
`run_test4`	Script file to run mcsi
`run_t_test4`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test4.86`	Full output file

E. Test run 5

Nonsymmetrized MCSI Hessian calculation on the HO – H – C_3H_7 system

This test run is an MCSI Hessian calculation for the HO – H – C_3H_7 system. This test run is the same as the test run 4 except in calculating generalized distances between points it uses the same internal coordinates as in Shepard interpolation (set s is the same as set **r**).

INPUT FILES:

`test5.81`	Configuration 1 input file
`test5.82`	Configuration 2 input file
`test5.83`	Electronic structure theory data file
`test5.84`	Geometry input file
`test5.85`	General input file
`param-test5.prm`	Modified tinker parameter file
`run_test5`	Script file to run mcsi
`run_t_test5`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test5.86`	Full output file

F. Test run 6

Nonsymmetrized MCSI Hessian calculation on the HO – H – C_3H_7 system

This test run is an MCSI Hessian calculation for the HO – H – C_3H_7 system. This test run is the same as the test run 5 except it uses thirty-nine internal coordinates in the Shepard interpolation step and three internal coordinates in calculating generalized distances (O–Ht, Ht–C, and O–C distances, where Ht represents the transferring hydrogen).

INPUT FILES:

`test6.81`	Configuration 1 input file
`test6.82`	Configuration 2 input file
`test6.83`	Electronic structure theory data file
`test6.84`	Geometry input file
`test6.85`	General input file
`param-test6.prm`	Modified tinker parameter file

`run_test6`	Script file to run mcsi
`run_t_test6`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test6.86`	Full output file

G. Test run 7

Nonsymmetrized SCMM1 Hessian calculation on the HO – H – C_3H_7 system

This test run is an SCMM1 Hessian calculation for the HO – H – C_3H_7 system. The SCMM1 option is explained in the section 6.5.1 on the ORDERCALC keyword. The configuration 1 has the atom connectivities equivalent to hydroxyl and propane. Note that in the `test7.85` input file, the RESONANCE section is not necessary and that in the MCENERGETICS section only ZERO1 keyword is used. This calculation provide an energy value relative to the molecular mechanics energy of the configuration 1 energy reference state (nonzero), and this value is different than one obtained directly by using `tinker`. The value of the parameter D (see Sec. VIII A 2) for this test run is 0.01.

INPUT FILES:

`test7.81`	Configuration 1 input file
`test7.84`	Geometry input file
`test7.85`	General input file
`param-test7.prm`	Modified tinker parameter file
`run_test7`	Script file to run mcsi
`run_t_test7`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test7.86`	Full output file

H. Test run 8

Nonsymmetrized SCMM2 Hessian calculation on the HO – H – C_3H_7 system

This test run is an SCMM2 Hessian calculation for the HO – H – C_3H_7 system. The SCMM2 option is explained in the section 6.5.1 on the ORDERCALC keyword. The configuration 2 has the atom connectivities equivalent to water and sec-propyl. Note that in the `test8.85` input file, the RESONANCE section is not necessary and that in the MCENERGETICS section only ZERO2 keyword is used. This calculation provide an energy value relative to the molecular mechanics energy of the configuration 2 energy reference state (not zero), and this value is different than one obtained directly by using TINKER. The value of the parameter D (see Sec. VIII A 2) for this test run is 0.01.

INPUT FILES:

`test8.82`	Configuration 2 input file
`test8.84`	Geometry input file
`test7.85`	General input file
`param-test8.prm`	Modified tinker parameter file
`run_test8`	Script file to run mcsi
`run_t_test8`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test8.86`	Full output file

I. Test run 9

Nonsymmetrized MCSI gradient calculation on the H_2N – H – CH_3 system

This test run is an MCSI gradient calculation for the H_2N – H – CH_3 system. The chemical reaction modeled in this calculation is $NH_2 + CH_4 \rightarrow NH_3 + CH_3$ with configuration 1 having the transferring hydrogen atom bonded to carbon and configuration 2 having this hydrogen atom bonded to nitrogen. This is one of the systems treated in Ref. 28 V_{12} is determined using Shepard interpolation with eleven electronic structure theory points and two SCMM points. The Shepard interpolation is carried out by using twenty-four internal coordinates, and the generalized distances are calculated using three internal coordinates. Set **s** includes one coordinate (N-H distance) which is not present in **r**. This type of coordinates is described in Sec. II B 1. Notice that set **r** includes the N-H-C angle (see Sec. II B 1). The JACOBIANS keyword is set to 1. The input geometry is given in mass-scaled bohrs (the scaling mass is 12.00 amu), and the printed results are mass-scaled also. The PRINTIC keyword is set to point, so the internal coordinates are printed for the point where MCSI calculation is carried out. The value of the parameter D (see Sec. VIII A 2) for this test run is 0.01.

INPUT FILES:

`test9.81`	Configuration 1 input file
`test9.82`	Configuration 2 input file
`test9.83`	Electronic structure theory data file
`test9.84`	Geometry input file
`test9.85`	General input file
`param-test9.prm`	Modified tinker parameter file
`run_test9`	Script file to run mcsi
`run_t_test9`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test9.86`	Full output file

J. Test run 10

Nonsymmetrized MCSI Hessian calculation on the H_2FC – H – CH_2Cl system

This test run is an MCSI Hessian calculation for the H_2FC – H – CH_2Cl system. The chemical reaction modeled in this calculation is CH_2F + CH_3Cl → CH_3F + CH_2Cl. Configuration 1 has the transferring hydrogen atom bonded to the carbon of CH_3Cl while configuration 2 has this hydrogen atom bonded to carbon of CH_3F. The electronic structure theory used is MPW1K/6-31+G(d,p), and the difference in energy between the energy reference states (optimized structures infinitely separated) is determined to be -1.32 kcal/mol. This is one of the systems treated in Ref. 28 V_{12} is determined using Shepard interpolation with fourteen points, twelve electronic structure theory points and two SCMM points. The Shepard interpolation is carried out by using twenty-seven internal coordinates. Two of these (C–Ht, and Ht–C distances, where Ht represents the transferring hydrogen) are used in calculating generalized distances. Note also that the molecular mechanics energy of the optimized CH_2F, CH_3Cl, CH_3F, and CH_2Cl structures are not zero so ZERO1 and ZERO2 keywords are included in MCENERGETICS section of the `esp.fu85` input file. The value of the parameter D (see Sec. VIII A 2) for this test run is 0.01.

INPUT FILES:

`test10.81`	Configuration 1 input file
`test10.82`	Configuration 2 input file
`test10.83`	Electronic structure theory data file
`test10.84`	Geometry input file
`test10.85`	General input file
`param-test10.prm`	Modified tinker parameter file
`run_test10`	Script file to run mcsi
`run_t_test10`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test10.86`	Full output file

K. Test run 11

Nonsymmetrized MCSI Hessian calculation on the H_2FC – H – CH_2Cl system

This test run is an MCSI Hessian calculation for the H_2FC – H – CH_2Cl system. This test run is the same as the test run 10 except in determining the resonance energy. The Shepard interpolation is not employed, and the resonance energy is determined through the expansion around POINT 2 of the `esp.fu83` input file.

INPUT FILES:

`test11.81`	Configuration 1 input file
`test11.82`	Configuration 2 input file
`test11.83`	Electronic structure theory data file
`test11.84`	Geometry input file
`test11.85`	General input file
`param-test11.prm`	Modified tinker parameter file
`run_test11`	Script file to run mcsi
`run_t_test11`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test11.86`	Full output file

L. Test run 12

Nonsymmetrized MCSI Hessian calculation on the H_2FC – H – CH_2Cl system

This test run is an MCSI Hessian calculation for the H_2FC – H – CH_2Cl system. This test run is the same as the test run 8 except in determining the resonance energy. In this test run the resonance energy is considered constant with a value given as the argument of keyword RESVALUE in the `esp.fu85` input file. Note that the input file `esp.fu83` is not used for this type of calculations.

INPUT FILES:

`test12.81`	Configuration 1 input file
`test12.82`	Configuration 2 input file
`test12.83`	Electronic structure theory data file
`test12.84`	Geometry input file
`test12.85`	General input file
`param-test12.prm`	Modified tinker parameter file
`run_test12`	Script file to run mcsi
`run_t_test12`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test12.86`	Full output file

M. Test run 13

Nonsymmetrized MCSI energy calculation on the C5H8 system

This test run is an MCSI energy calculation for the 1,3-cis-pentadiene system. The chemical process modeled in this calculation is the isomerization of 1,3-cis-pentadiene, in which configuration 1 has the transferring hydrogen atom bonded to carbon 5 while configuration 2 has this hydrogen atom bonded to carbon 6. This is one of the systems treated in Ref. 3 Both the configuration 1 energy reference state and the configuration 2 energy reference state

are 1,3-cis-pentadiene in its equilibrium geometry (s-trans conformation) determined using tinker with a VESCF calculation. The parameter file param-test13.prm is a modified version of the standard MM3 force field parameters and includes the VESCF-optimized parameters. A new atom type, which is labeled atom type number 20, is defined to use the VESCF-optimized parameters and to avoid the VESCF calculations (see Sec. VIII A 3). No further special treatment for the π-system is made. The calculation uses the MORSE bond type for the C(sp3)H bonds (the bond strength of the C(sp3)-H bond is taken as 99 kcal/mol) and the TAYLOR bond type for all other bonds in the molecule. The electronic structure data is obtained from AM1 calculations.

V_{12} is determined using Shepard interpolation with five points, in particular three electronic structure theory points and two SCMM points. The two SCMM points are 1,3-cis-pentadiene in the s-cis conformation. (Note that the structure, energy, and frequencies for the s-cis conformation of pentadiene are exactly the same as those using the VESCF method with the original set of parameters.) The Shepard interpolation is carried out by using forty-three internal coordinates, with three of these used in calculating the generalized distance between points. The `esp.fu84` input file contains two geometries (given in unscaled angstroms), and the MCSI energy calculation is carried out for both of these geometries. The PRINTIC keyword is set to all so the internal coordinates are printed for all the Shepard points and for the two points where MCSI calculations are carried out. The value of the parameter D (see Sec. VIII A 2) for this test run is 0.01.

INPUT FILES:

`test13.81`	Configuration 1 input file
`test13.82`	Configuration 2 input file
`test13.83`	Electronic structure theory data file
`test13.84`	Geometry input file
`test13.85`	General input file
`param-test13.prm`	Modified tinker parameter file
`run_test13`	Script file to run mcsi
`run_t_test13`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test13.86`	Full output file

N. Test run 14

Symmetrized MCSI gradient calculation on the OH + H_2 system. All three hydrogen atoms are treated as indistinguishable.

This test run is an MCSI Hessian calculation for the OH + H_2 system. The two MM configurations used to represent reactants and products are H_2O + H and OH + H_2, respectively. The input file esp.fu81 specifies MM configuration 1. Input files `esp.fu72--esp.fu76` specify the five equivalent MM configurations. Input files `esp.fu82` and `esp.fu62--esp.fu66` contain similar information for MM configuration 2. We used three electronic-structure Shepard points placed at (i) the saddle point of the H-transfer reaction, (ii) the saddle point of the exchange reaction, and (iii) the ammonia-like equilibrium structure OH_3. These three unique Shepard points are specified in the `esp.fu83` file. The other 15 data points that are symmetrically equivalent to points (i), (ii), and (iii) and are also used in the interpolation, are generated by the program after reading the information from the `esp.fu83` file. The electronic structure level is MPW1K/6-31+G(d,p). In addition to the three electronic structure Shepard points, there are two SCMM points, which are the van der Waals wells, $OH\cdots H_2$ and $H_2O\cdots H$. The set $\mathbf{r}$ of internal coordinates includes all six internuclear distances and all twelve bond angles. The set $\mathbf{s}$ consists of the six internuclear distances. It is essential that both sets $\mathbf{r}$ and $\mathbf{s}$ include, in addition to the internal coordinates that one would use in a nonsymmetrized MCSI calculation, all symmetrically equivalent coordinates. For example, if one wants to include the O–H1 distance to either of these sets, one should also include the O–H_2 and O–H_3 distances because these hydrogen atoms are symmetrically equivalent. The SYMMETRY section in the `esp.fu85` file specifies parameters α and σ that will be used to generate symmetric SCMM potentials for configurations 1 and 2. The `esp.84` file contains an arbitrary geometry followed by the five equivalent geometries with the permuted Cartesian coordinates for the hydrogen atoms. The output file contains the symmetrized MCSI energies, and gradient vectors. Note that V_{nn}, V_{12}, and the lowest eigenvalue V of the matrix $\mathbf{V}$ are scalar, and all these values are the same for all six input geometries. The gradient vectors have the same components for all six geometries, but these components are arranged in different order corresponding to the six different permutations.

INPUT FILES:

`test14.81`	Configuration 1; permutation (E)
`test14.72`	Configuration 1; permutation (12)
`test14.73`	Configuration 1; permutation (23)
`test14.74`	Configuration 1; permutation (13)
`test14.75`	Configuration 1; permutation (123)
`test14.76`	Configuration 1; permutation (132)
`test14.82`	Configuration 2; permutation (E)
`test14.62`	Configuration 2; permutation (12)
`test14.63`	Configuration 2; permutation (23)
`test14.64`	Configuration 2; permutation (13)
`test14.65`	Configuration 2; permutation (123)
`test14.66`	Configuration 2; permutation (132)
`test14.83`	Electronic structure theory data file
`test14.84`	Geometry input file
`test14.85`	General input file
`param-test14.prm`	Modified tinker parameter file
`run_test14`	Script file to run mcsi
`run_t_test14`	Script file to run mcsi and to time

the run

OUTPUT FILES:

`test14.86` Full output file

Example of a set of $m!$ input files that specify the MM configurations of reactants ($OH + H_2$) is given below. Note that these files contain the same Cartesian coordinates but different connectivity patters that result from applying each PMM to the atom types and connectivities given in the `test14.fu81` file.

```
test14.81
4 reactant well  (E)
1  O    3.195183      .105829     -.396280    20     2
2  H    4.162021      .189153     -.416404    21     1
3  H     .000000      .000000      .000000   122     4
4  H     .015372     -.337746     -.659867   122     3

test14.72
4 reactant well  (12)
1  O    3.195183      .105829     -.396280    20     3
2  H    4.162021      .189153     -.416404   122     4
3  H     .000000      .000000      .000000    21     1
4  H     .015372     -.337746     -.659867   122     2

test14.73
4 reactant well  (23)
1  O    3.195183      .105829     -.396280    20     2
2  H    4.162021      .189153     -.416404    21     1
3  H     .000000      .000000      .000000   122     4
4  H     .015372     -.337746     -.659867   122     3

test14.74
4 reactant well  (13)
1  O    3.195183      .105829     -.396280    20     4
2  H    4.162021      .189153     -.416404   122     3
3  H     .000000      .000000      .000000   122     2
4  H     .015372     -.337746     -.659867    21     1

test14.75
4 reactant well  (123)
1  O    3.195183      .105829     -.396280    20     3
2  H    4.162021      .189153     -.416404   122     4
3  H     .000000      .000000      .000000    21     1
4  H     .015372     -.337746     -.659867   122     2

test14.76
4 reactant well  (132)
1  O    3.195183      .105829     -.396280    20     4
2  H    4.162021      .189153     -.416404   122     3
3  H     .000000      .000000      .000000   122     2
4  H     .015372     -.337746     -.659867    21     1
```

The MCSI output file `test14.86` contains the symmetrized matrix elements (SCMM1, SCMM2, and the resonance energy), the MCSI energies, and the symmetrized gradient vectors.

```
(E)
                              hartree         eV        cm**-1         kcal
  SCMM1     energy (V11)       .0519673     1.414115    11405.512     32.61000
  SCMM2     energy (V22)       .2202130     5.992349    48331.156    138.18572
  Resonance energy (V12)       .0202639      .551414     4447.418     12.71581
  MCSI      energy (V)         .0495611     1.348638    10877.406     31.10007

 ** Cartesian Coordinates (unscaled angstroms):
```

```
 -2.25480000E-01  5.90000000E-02  0.00000000E+00 -3.67630000E-01 -8.98670000E-01
  0.00000000E+00  1.03452000E+00  6.08900000E-02  0.00000000E+00  1.59161000E+00
 -1.00530000E-01  0.00000000E+00
 ** Cartesian Gradient (hartrees per unscaled angstrom):
  8.32028495E-02 -8.43227395E-03  0.00000000E+00 -2.32641931E-02 -2.72840556E-03
  0.00000000E+00  2.65507406E-01 -9.69894223E-02  0.00000000E+00 -3.25446063E-01
  1.08150102E-01  0.00000000E+00

(12)
                              hartree         eV        cm**-1         kcal
  SCMM1     energy (V11)      .0519673     1.414115    11405.512     32.61000
  SCMM2     energy (V22)      .2202130     5.992349    48331.156    138.18572
  Resonance energy (V12)      .0202639      .551414     4447.418     12.71581
  MCSI      energy (V)        .0495611     1.348638    10877.406     31.10007

 ** Cartesian Coordinates (unscaled angstroms):
 -2.25480000E-01  5.90000000E-02  0.00000000E+00  1.03452000E+00  6.08900000E-02
  0.00000000E+00 -3.67630000E-01 -8.98670000E-01  0.00000000E+00  1.59161000E+00
 -1.00530000E-01  0.00000000E+00
 ** Cartesian Gradient (hartrees per unscaled angstrom):
  8.32028495E-02 -8.43227395E-03  0.00000000E+00  2.65507406E-01 -9.69894223E-02
  0.00000000E+00 -2.32641931E-02 -2.72840556E-03  0.00000000E+00 -3.25446063E-01
  1.08150102E-01  0.00000000E+00

(23)
                              hartree         eV        cm**-1         kcal
  SCMM1     energy (V11)      .0519673     1.414115    11405.512     32.61000
  SCMM2     energy (V22)      .2202130     5.992349    48331.156    138.18572
  Resonance energy (V12)      .0202639      .551414     4447.418     12.71581
  MCSI      energy (V)        .0495611     1.348638    10877.406     31.10007

 ** Cartesian Coordinates (unscaled angstroms):
 -2.25480000E-01  5.90000000E-02  0.00000000E+00 -3.67630000E-01 -8.98670000E-01
  0.00000000E+00  1.59161000E+00 -1.00530000E-01  0.00000000E+00  1.03452000E+00
  6.08900000E-02  0.00000000E+00
 ** Cartesian Gradient (hartrees per unscaled angstrom):
  8.32028495E-02 -8.43227395E-03  0.00000000E+00 -2.32641931E-02 -2.72840556E-03
  0.00000000E+00 -3.25446063E-01  1.08150102E-01  0.00000000E+00  2.65507406E-01
 -9.69894223E-02  0.00000000E+00

(13)
                              hartree         eV        cm**-1         kcal
  SCMM1     energy (V11)      .0519673     1.414115    11405.512     32.61000
  SCMM2     energy (V22)      .2202130     5.992349    48331.156    138.18572
  Resonance energy (V12)      .0202639      .551414     4447.418     12.71581
  MCSI      energy (V)        .0495611     1.348638    10877.406     31.10007

 ** Cartesian Coordinates (unscaled angstroms):
 -2.25480000E-01  5.90000000E-02  0.00000000E+00  1.59161000E+00 -1.00530000E-01
  0.00000000E+00  1.03452000E+00  6.08900000E-02  0.00000000E+00 -3.67630000E-01
 -8.98670000E-01  0.00000000E+00
 ** Cartesian Gradient (hartrees per unscaled angstrom):
  8.32028495E-02 -8.43227395E-03  0.00000000E+00 -3.25446063E-01  1.08150102E-01
  0.00000000E+00  2.65507406E-01 -9.69894223E-02  0.00000000E+00 -2.32641931E-02
 -2.72840556E-03  0.00000000E+00

(123)
                              hartree         eV        cm**-1         kcal
  SCMM1     energy (V11)      .0519673     1.414115    11405.512     32.61000
  SCMM2     energy (V22)      .2202130     5.992349    48331.156    138.18572
```

```
 Resonance energy (V12)      .0202639      .551414     4447.418     12.71581
 MCSI      energy (V)        .0495611     1.348638    10877.406     31.10007

** Cartesian Coordinates (unscaled angstroms):
-2.25480000E-01  5.90000000E-02  0.00000000E+00  1.59161000E+00 -1.00530000E-01
 0.00000000E+00 -3.67630000E-01 -8.98670000E-01  0.00000000E+00  1.03452000E+00
 6.08900000E-02  0.00000000E+00
** Cartesian Gradient (hartrees per unscaled angstrom):
 8.32028495E-02 -8.43227395E-03  0.00000000E+00 -3.25446063E-01  1.08150102E-01
 0.00000000E+00 -2.32641931E-02 -2.72840556E-03  0.00000000E+00  2.65507406E-01
-9.69894223E-02  0.00000000E+00

(132)
                              hartree         eV        cm**-1         kcal
 SCMM1     energy (V11)      .0519673     1.414115    11405.512     32.61000
 SCMM2     energy (V22)      .2202130     5.992349    48331.156    138.18572
 Resonance energy (V12)      .0202639      .551414     4447.418     12.71581
 MCSI      energy (V)        .0495611     1.348638    10877.406     31.10007

** Cartesian Coordinates (unscaled angstroms):
-2.25480000E-01  5.90000000E-02  0.00000000E+00  1.03452000E+00  6.08900000E-02
 0.00000000E+00  1.59161000E+00 -1.00530000E-01  0.00000000E+00 -3.67630000E-01
-8.98670000E-01  0.00000000E+00
** Cartesian Gradient (hartrees per unscaled angstrom):
 8.32028495E-02 -8.43227395E-03  0.00000000E+00  2.65507406E-01 -9.69894223E-02
 0.00000000E+00 -3.25446063E-01  1.08150102E-01  0.00000000E+00 -2.32641931E-02
-2.72840556E-03  0.00000000E+00
```

O. Test run 15

Symmetrized MCSI gradient calculation on the OH + H_2 system. All three hydrogen atoms are treated as indistinguishable.

This test run is the same as test 14 except that the sets of internal coordinates **r** and **s** are the same, and they both consist of the six internuclear distances.

INPUT FILES:

`test15.81`	Configuration 1; permutation (E)
`test15.72`	Configuration 1; permutation (12)
`test15.73`	Configuration 1; permutation (23)
`test15.74`	Configuration 1; permutation (13)
`test15.75`	Configuration 1; permutation (123)
`test15.76`	Configuration 1; permutation (132)
`test15.82`	Configuration 2; permutation (E)
`test15.62`	Configuration 2; permutation (12)
`test15.63`	Configuration 2; permutation (23)
`test15.64`	Configuration 2; permutation (13)
`test15.65`	Configuration 2; permutation (123)
`test15.66`	Configuration 2; permutation (132)
`test15.83`	Electronic structure theory data file
`test15.84`	Geometry input file
`test15.85`	General input file
`param-test15.prm`	Modified tinker parameter file

`run_test15`	Script file to run mcsi
`run_t_test15`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test15.86`	Full output file

P. Test run 16

Nonsymmetrized EE-MM energy and charge calculation on the CH_3Cl system

This test run is an EE-MM energy and charge calculation with the SCMM1 option for the CH_3Cl system. The EE-MM calculation is carried out using eqs. 2.18 and 2.19. Note that in the `test16.85` input file, the ICSHEPARD keyword in the RESONANCE section is required to transform the derivatives of the charges with respect to the Cartesian coordinates to those with respect to the internal coordinates, and that in the MCENERGETICS section only the ZERO1 keyword is used.

INPUT FILES:

`test16.81`	Configuration 1 input file
`test16.84`	Geometry input file
`test16.85`	General input file
`param-test16.prm`	Modified tinker parameter file
`run_test16`	Script file to run mcsi
`run_t_test16`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test16.86`	Full output file

Q. Test run 17

Nonsymmetrized EE-MCSI Hessian and CRK calculation on the $Cl^- + CH_3Cl'$ system

This test run is an EE-MCSI Hessian and CRK calculation for the $Cl^- + CH_3Cl'$ system. This system is treated in Ref. 6. Configuration 1 corresponds to the $Cl^- + CH_3Cl'$ system while configuration 2 corresponds to $ClCH_3 + Cl'^-$ the system. The electronic structure theory used is MPW1K, and the basis set is 6-31G(d,p) for C and H atoms and 6-31+G(d,p) for Cl. The potential barrier is determined to be 3.2 kcal/mol in gas phase. V_{12} is determined using Shepard interpolation with three stationary electronic structure theory points (the precursor ion-dipole complex, the saddle point, and the successor ion-dipole complex) and nine nonstationary electronic structure theory points. The Shepard interpolation is carried out by using fifteen internal coordinates (set `r`). Three of these (C–Cl, C–Cl′, and Cl–Cl′) are included in the set **s** that is used to calculate the generalized distance.

INPUT FILES:

`test17.81`	Configuration 1 input file
`test17.82`	Configuration 2 input file
`test17.83`	Electronic structure theory data file
`test17.84`	Geometry input file
`test17.85`	General input file
`param-test17.prm`	Modified tinker parameter file
`run_test17`	Script file to run mcsi
`run_t_test17`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test17.86`	Full output file

Output test17.86 file (in part)

```
 ********************************  Point   2  ********************************

 Atom Number   Atomic Symbol      X(angstrom)     Y(angstrom)     Z(angstrom)
     1               C             .00000000       .00000000       .00000000
     2               H            -.53435591       .92553159       .00000000
     3               H            -.53435591      -.92553159       .00000000
     4               H            1.06871182       .00000000       .00000000
     5               Cl            .00000000       .00000000      2.31503001
     6               Cl            .00000000       .00000000     -2.31503001

                               hartree         eV        cm**-1         kcal
  SCMM1     energy (V11)      .0865608     2.355460    18997.910     54.31776
  SCMM2     energy (V22)      .0865608     2.355460    18997.910     54.31776
  Resonance energy (V12)      .0814768     2.217115    17882.088     51.12746
  MCSI      energy (V)        .0050841      .138345     1115.822      3.19030

------------------------------  EE-MCSI Results  ------------------------------

 Atom Number   Atomic Symbol        Phi(a.u.)       Phi(V)        Charge(a.u.)
     1               C             .00000000       .00000000      -.02600043
     2               H             .00000000       .00000000       .11851415
     3               H             .00000000       .00000000       .11851415
     4               H             .00000000       .00000000       .11851415
     5               Cl            .00000000       .00000000      -.66477100
     6               Cl            .00000000       .00000000      -.66477100

                               hartree         eV        cm**-1      kcal/mol
  V(QM)           energy       .0050841      .138345     1115.822      3.19030
  V(QM/MM)        energy       .0000000      .000000        .000       .00000
  V(QM)+V(QM/MM) energy        .0050841      .138345     1115.822      3.19030

********************************  Point   4  ********************************

 Atom Number   Atomic Symbol      X(angstrom)     Y(angstrom)     Z(angstrom)
     1               C             .00000000       .00000000       .00000000
     2               H            -.53520866       .92700860       .00000000
     3               H            -.53520866      -.92700860       .00000000
     4               H            1.07041733       .00000000       .00000000
     5               Cl            .00000000       .00000000      2.31589993
     6               Cl            .00000000       .00000000     -2.31589993

                               hartree         eV        cm**-1         kcal
```

```
 SCMM1     energy (V11)     -.1159039    -3.153932   -25437.965    -72.73080
 SCMM2     energy (V22)     -.1159039    -3.153932   -25437.965    -72.73080
 Resonance energy (V12)      .0783440     2.131867    17194.528     49.16163
 MCSI      energy (V)       -.1942479    -5.285799   -42632.492   -121.89242

------------------------------  EE-MCSI Results  ------------------------------

Atom Number   Atomic Symbol        Phi(a.u.)       Phi(eV)       Charge(a.u.)
    1               C              .17460304      4.75122986      -.01524567
    2               H              .16932528      4.60761341       .12671321
    3               H              .16932528      4.60761341       .12671321
    4               H              .16932528      4.60761341       .12671321
    5               Cl             .19151678      5.21148000      -.68244699
    6               Cl             .19151678      5.21148000      -.68244699

                                  hartree         eV        cm**-1      kcal/mol
 V(QM)          energy          .0054469       .148218     1195.448      3.41796
 V(QM/MM)       energy         -.1996948     -5.434017   -43827.940   -125.31038
 V(QM)+V(QM/MM) energy         -.1942479     -5.285799   -42632.492   -121.89242
```

R. Test run 18

This test run uses the input from Test 9 (nonsymmetrized MCSI calculation on the $H_2N - H - CH_3$ system), but it prints out the Hessian matrix calculated numerically and analytically (HESSTEST option).

```
   Numerical      Analytical    Difference
  .3506667179    .3506672724   -.0000005545
 -.0862289060   -.0862289218    .0000000159
  .0646117881    .0646123401   -.0000005519
 -.0000001388   -.0000001058   -.0000000330
  .0000001388    .0000001865   -.0000000477
  .0458322269    .0458328313   -.0000006044
 -.3306099491   -.3306099384   -.0000000107
```

INPUT FILES:

`test18.81`	Configuration 1 input file
`test18.82`	Configuration 2 input file
`test18.83`	Electronic structure theory data file
`test18.84`	Geometry input file
`test18.85`	General input file
`param-test18.prm`	Modified tinker parameter file
`run_test18`	Script file to run mcsi
`run_t_test18`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test18.86`	Full output file

S. Test run 19

This test run uses the input from Test run 15 (symmetrized MCSI calculation on the OH + H_2 system), except that it uses the GRADTEST option in the MCGENERAL section to print the gradient vector calculated numerically and analytically.

```
          Numerical Gradient
 .0479849099 -.0033938573 .0000000000
-.0081293623 -.0009467760 .0000000000
 .1376129581 -.0508624819 .0000000000
-.1774685054  .0552031155 .0000000000

         Analytical Gradient
 .0479849098 -.0033938573 .0000000000
-.0081293624 -.0009467760 .0000000000
 .1376129584 -.0508624820 .0000000000
-.1774685058  .0552031153 .0000000000
```

INPUT FILES:

`test19.81`	Configuration 1; permutation (E)
`test19.72`	Configuration 1; permutation (12)
`test19.73`	Configuration 1; permutation (23)
`test19.74`	Configuration 1; permutation (13)
`test19.75`	Configuration 1; permutation (123)
`test19.76`	Configuration 1; permutation (132)
`test19.82`	Configuration 2; permutation (E)
`test19.62`	Configuration 2; permutation (12)
`test19.63`	Configuration 2; permutation (23)
`test19.64`	Configuration 2; permutation (13)
`test19.65`	Configuration 2; permutation (123)
`test19.66`	Configuration 2; permutation (132)
`test19.83`	Electronic structure theory data file
`test19.84`	Geometry input file
`test19.85`	General input file
`param-test19.prm`	Modified tinker parameter file
`run_test19`	Script file to run mcsi
`run_t_test19`	Script file to run mcsi and to time the run

OUTPUT FILES:
`test19.86` Full output file

T. Test run 20

This test run is for the symmetrized MCSI PES of the OH + H_2 system using only gradient information.

INPUT FILES:

`test20.81`	Configuration 1; permutation (E)
`test20.72`	Configuration 1; permutation (12)
`test20.73`	Configuration 1; permutation (23)
`test20.74`	Configuration 1; permutation (13)
`test20.75`	Configuration 1; permutation (123)
`test20.76`	Configuration 1; permutation (132)
`test20.82`	Configuration 2; permutation (E)
`test20.62`	Configuration 2; permutation (12)
`test20.63`	Configuration 2; permutation (23)
`test20.64`	Configuration 2; permutation (13)
`test20.65`	Configuration 2; permutation (123)
`test20.66`	Configuration 2; permutation (132)
`test20.83`	Electronic structure theory data file
`test20.84`	Geometry input file
`test20.85`	General input file
`param-test20.prm`	Modified tinker parameter file
`run_test20`	Script file to run mcsi
`run_t_test20`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test20.86`	Full output file

U. Test run 21

This test run is for the MCSI PES of the 5,7,8-trimethyl-croman-6-ol + CH_3 reaction using only gradient information.

INPUT FILES:

`test21.81`	Configuration 1 input file
`test21.82`	Configuration 2 input file
`test21.83`	Electronic structure theory data file
`test21.84`	Geometry input file
`test21.85`	General input file
`param-test21.prm`	Modified tinker parameter file
`run_test21`	Script file to run mcsi
`run_t_test21`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test21.86`	Full output file

V. Test run 22

This test run uses the input from Test run 17 (non-symmetrized EE-MCSI calculation on the Cl^- + CH_3Cl' system), except that it uses the CHARGETEST option in the MCGENERAL section to print the charges calculated numerically and analytically.

INPUT FILES:

`test22.81`	Configuration 1 input file
`test22.82`	Configuration 2 input file
`test22.83`	Electronic structure theory data file
`test22.84`	Geometry input file
`test22.85`	General input file
`param-test22.prm`	Modified tinker parameter file
`run_test22`	Script file to run mcsi
`run_t_test22`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test22.86`	Full output file

W. Test run 23

This test run uses the input from Test run 17 (non-symmetrized EE-MCSI calculation on the Cl^- + CH_3Cl' system), except that it uses the DQDRTEST option in the MCGENERAL section to print the CRK $\boldsymbol{\kappa} = \dfrac{\partial \mathbf{Q}}{\partial \mathbf{R}}$ calculated numerically and analytically.

INPUT FILES:

`test23.81`	Configuration 1 input file
`test23.82`	Configuration 2 input file
`test23.83`	Electronic structure theory data file
`test23.84`	Geometry input file
`test23.85`	General input file
`param-test23.prm`	Modified tinker parameter file
`run_test23`	Script file to run mcsi
`run_t_test23`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test23.86`	Full output file

X. Test run 24

This test run uses the input from Test run 17 (non-symmetrized EE-MCSI calculation on the Cl^- + CH_3Cl'

system), except that it uses the DQDRTEST option in the MCGENERAL section to print the CRK $\boldsymbol{\chi} = \frac{\partial \mathbf{Q}}{\partial \boldsymbol{\Phi}}$ calculated numerically and analytically.

INPUT FILES:

`test24.81`	Configuration 1 input file
`test24.82`	Configuration 2 input file
`test24.83`	Electronic structure theory data file
`test24.84`	Geometry input file
`test24.85`	General input file
`param-test24.prm`	Modified tinker parameter file
`run_test24`	Script file to run mcsi
`run_t_test24`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test24.86`	Full output file

Y. Test run 25

Nonsymmetrized EE-MCSI energy calculation on the CH_3COO^- + CH_2ClCH_2Cl system

This test run is an EE-MCSI energy calculation for the CH_3COO^- + CH_2ClCH_2Cl system. This system is treated in Ref. 36. Configuration 1 corresponds to the CH_3COO^- + CH_2ClCH_2Cl system while configuration 2 corresponds to $CH_3COOCH_2CH_2Cl$ + Cl^- the system. The electronic structure theory used is MPW1K, and the basis set is 6-31G(d,p) for C and H atoms and 6-31+G(d,p) for O and Cl. The Shepard points are determined based on the QM/MM potential energy surface in haloalkane dehalogenase. V_{12} is determined using Shepard interpolation with three electronic structure theory points (the reactant, the saddle point, and the product) and five nonstationary electronic structure theory points. The Shepard interpolation is carried out by using 66 internal coordinates (set `r`), where sines of dihedral angles are used for the Taylor expansion (See Sec. V). Three of these (C–O, C–Cl, and C–O–C–Cl) are included in the set **s** that is used to calculate the generalized distance.

INPUT FILES:

`test25.81`	Configuration 1 input file
`test25.82`	Configuration 2 input file
`test25.83`	Electronic structure theory data file
`test25.84`	Geometry input file
`test25.85`	General input file
`param-test25.prm`	Modified tinker parameter file
`run_test25`	Script file to run mcsi
`run_t_test25`	Script file to run mcsi and to time the run

OUTPUT FILES:

`test25.86`	Full output file

XII. COMPUTERS AND OPERATING SYSTEMS ON WHICH THE CODE HAS BEEN DEVELOPED AND TESTED

The computers, operating systems, and compiler versions on which various versions of MCSI have been tested are listed in Tables 10.1 and 10.2. The compiler and loader commands used for testing the code are listed in Tables 10.3 and 10.4. Note that the FORTRAN compiler on the Compaq computer uses a Compaq FORTRAN 90 compiler (which is called f77) with several compilation flags for FORTRAN 77.

TABLE I: Operating systems on the various machines on which various versions of the code have been tested

Version	Machine	Operating system
1.0	IBM SP Power3	AIX 4.3.4.0
	SGI Origin 3800 R14000	IRIX 6.5.12f
	Compaq ES40 Alpha 500	Tru64 4.0f
1.0.1	IBM SP Power3	AIX 4.3
	IBM Regatta Power4	AIX 5.1
	SGI Altix 3000 Linux SMP	2.4.20
	Netfinity Linux Cluster (Intel Pentium III)	RedHat Linux 7.2 – 2.4.9 kernel
1.1	IBM SP Power3	AIX 4.3
	IBM Regatta Power4	AIX 5.1
	SGI Altix 3000	Linux SMP 2.4.20
	Netfinity Linux Cluster (Intel Pentium III)	RedHat Linux 7.2 – 2.4.9 kernel
1.1.1	IBM Regatta Power4	AIX 5.2
2007	IBM Regatta Power4	AIX 5.2
	SGI Altix 3000	SuSE Linux

	IBM BladeCenter Linux Cluster	SuSE Linux
2008	IBM Regatta Power4	AIX 5.3
	SGI Altix 3000	SuSE Linux Enterprise Server 9 (ia64) – 2.6.5 kernel
	IBM BladeCenter Linux Cluster	SuSE Linux Enterprise Server 9 (x86_64) – 2.6.5 kernel
20082	IBM Regatta Power4	AIX 5.3
2009	IBM Regatta Power4	AIX 5.3
	SGI Altix 3000	SuSE Linux Enterprise Server 9 (ia64) – 2.6.5 kernel
	IBM BladeCenter Linux Cluster	SuSE Linux Enterprise Server 9 (x86_64) – 2.6.5 kernel
	SGI Altix XE 1300 Linux Cluster	Linux
2009-1	IBM BladeCenter Linux Cluster	SuSE Linux Enterprise Server 9 (x86_64) – 2.6.5 kernel
2010-1	Linux Cluster	CentOS 4.4 (x86_64) – 2.6.9 kernel

TABLE II: Compiler versions used for various machines

Version	Machine	Compiler version
1.0	IBM SP Power3	XL FORTRAN 6.1.0.3
	SGI Origin 3800 R14000	MIPSpro 7.3.1.2m
	Compaq ES40 Alpha 500	Compaq FORTRAN V5.4-1283-46ABA
1.0.1	IBM SP Power3	XL FORTRAN 6.1.0.3
	IBM Regatta Power4	XL FORTRAN 7.1.1.2
	SGI Altix 3000	Intel C/C++ and Fortran compilers 7.0
	Netfinity Linux Cluster (Intel Pentium III)	Portland Group FORTRAN 77 compiler 5.0
1.1	IBM SP Power3	XL FORTRAN 6.1.0.3
	IBM Regatta Power4	XL FORTRAN 7.1.1.2
	SGI Altix 3000	Intel C/C++ and Fortran compilers 7.0
	Netfinity Linux Cluster (Intel Pentium III)	Portland Group FORTRAN 77 compiler 5.0
1.1.1	IBM Regatta Power4	XL Fortran for AIX
2007	IBM Regatta Power4	XL Fortran for AIX
	SGI Altix 3000	g77 version 3.4.6
	IBM BladeCenter Linux Cluster	Portland Group Fortran 90 compiler version 6.2-5
	IBM BladeCenter Linux Cluster	g77 version 3.4.6
2008	IBM Regatta Power4	XL Fortran for AIX version 10.1
	SGI Altix 3000	g77 version 3.3.3
	SGI Altix 3000	Intel Fortran complier version 8.1
	IBM BladeCenter Linux Cluster	Portland Group Fortran 90 compiler version 6.2-5
	IBM BladeCenter Linux Cluster	g77 version 3.3.3
	IBM BladeCenter Linux Cluster	Intel Fortran complier version 9.1
20082	IBM Regatta Power4	XL Fortran for AIX version 10.1
2009	IBM Regatta Power4	XL Fortran for AIX version 10.1
	SGI Altix 3000	g77 version 3.3.3
	SGI Altix 3000	Intel Fortran complier version 8.1
	IBM BladeCenter Linux Cluster	Portland Group Fortran 90 compiler version 6.2-5
	IBM BladeCenter Linux Cluster	g77 version 3.3.3
	IBM BladeCenter Linux Cluster	Intel Fortran complier version 9.1
	SGI Altix XE 1300 Linux Cluster	Intel Fortran complier version 11
2009-1	IBM BladeCenter Linux Cluster	Intel Fortran complier version 9.1
2010-1	Linux Cluster	Intel Fortran complier version 11.0
	Linux Cluster	GNU Fortran complier version 4.1.0
	Linux Cluster	PGI Fortran complier version 7.1-2

TABLE III: Compiler options used for various machines

Version	Machine	Compiler commands
1.0	IBM SP Power3	xlf -c -qdpc -qmaxmem=-1
	SGI Origin 3800 R14000	f77 -c -O3 -static -64
	Compaq ES40 Alpha 500	f77 -c -O3
1.0.1	IBM SP Power3	xlf -c -qdpc -qmaxmem=-1
	IBM Regatta Power4	xlf -c -qdpc -qmaxmem=-1
	SGI Altix 3000	efc -c -static -w
	Netfinity Linux Cluster (Intel Pentium III)	pgf77 -c -O3
1.1	IBM SP Power3	xlf -c -qdpc -qmaxmem=-1
	IBM Regatta Power4	xlf -c -qdpc -qmaxmem=-1
	SGI Altix 3000	efc -c -static -w
	Netfinity Linux Cluster (Intel Pentium III)	pgf77 -c -O3
1.1.1	IBM Regatta Power4	xlf -c -qdpc -qmaxmem=-1
2007	IBM Regatta Power4	xlf -c -qdpc -qmaxmem=-1
	SGI Altix 3000	g77 -c -O3
	IBM BladeCenter Linux Cluster	pgf90 -c -Bstatic -Msave -fast
	IBM BladeCenter Linux Cluster	g77 -c -O3
2008	IBM Regatta Power4	xlf -c -qdpc -qmaxmem=-1
	SGI Altix 3000	g77 -c -O3
	SGI Altix 3000	ifort -c -w
	IBM BladeCenter Linux Cluster	pgf90 -c -Bstatic -Msave -fast
	IBM BladeCenter Linux Cluster	g77 -c -O3
	IBM BladeCenter Linux Cluster	ifort -c -w
20082	IBM Regatta Power4	xlf -c -qdpc -qmaxmem=-1
2009	IBM Regatta Power4	xlf -g -c -qdpc -O3 -qmaxmem=-1
	SGI Altix 3000	g77 -c -O3
	SGI Altix 3000	ifort -c -w
	IBM BladeCenter Linux Cluster	pgf90 -c -Bstatic -Msave -fast
	IBM BladeCenter Linux Cluster	g77 -c -O3
	IBM BladeCenter Linux Cluster	ifort -c -w
	SGI Altix XE 1300 Linux Cluster	ifort -O3 -c
2009-1	IBM BladeCenter Linux Cluster	ifort -c -w
2010-1	Linux Cluster	ifort -c -w
	Linux Cluster	gfortran -c -w
	Linux Cluster	pgf77 -c -w

TABLE IV: Loader commands used for various machines

Version	Machine	Loader commands
1.0	IBM SP Power3	xlf -o
	SGI Origin 3800 R14000	f77 -64 -o
	Compaq ES40 Alpha 500	f77 -o
1.0.1	IBM SP Power3	xlf -o
	IBM Regatta Power4	xlf -o
	SGI Altix 3000	efc -Vaxlib -o
	Netfinity Linux Cluster (Intel Pentium III)	pgf77 -o
1.1	IBM SP Power3	xlf -o
	IBM Regatta Power4	xlf -o
	SGI Altix 3000	efc -Vaxlib -o
	Netfinity Linux Cluster (Intel Pentium III)	pgf77 -o
1.1.1	IBM Regatta Power4	xlf -o
2007	IBM Regatta Power4	xlf -o
	SGI Altix 3000	g77-o
	IBM BladeCenter Linux Cluster	pgf90 -o

	IBM BladeCenter Linux Cluster	g77 -o
2008	IBM Regatta Power4	xlf -o
	SGI Altix 3000	g77 -o
	IBM BladeCenter Linux Cluster	pgf90 -o
	IBM BladeCenter Linux Cluster	g77 -o
	IBM BladeCenter Linux Cluster	ifort -o
20082	IBM Regatta Power4	xlf -o
2009	IBM Regatta Power4	xlf -o
	SGI Altix 3000	g77 -o
	IBM BladeCenter Linux Cluster	pgf90 -o
	IBM BladeCenter Linux Cluster	g77 -o
	IBM BladeCenter Linux Cluster	ifort -o
	SGI Altix XE 1300 Linux Cluster	ifort -o
2009-1	IBM BladeCenter Linux Cluster	ifort -o
2010-1	Linux Cluster	ifort -o
	Linux Cluster	gfortran -o
	Linux Cluster	pgf77 -o

XIII. REVISION HISTORY

A. mn versions of TINKER

In this section we will list the changes made with respect to the original TINKER 3.5 release of the code by Jay W. Ponder.

1. TINKER–*version 3.5mn1 (December 1999)*

TINKER–VERSION3.5MN1 is the first mn version of the tinker program. The differences between TINKER 3.5 and tinker 3.5mn1 are:

- Two bugs found in the original tinker 3.5 version have been corrected. These bugs are also corrected in later TINKER versions distributed by J. W. Ponder. These bugs are:
  - When the BUCKINGHAM option is selected for the van der Waals energy term, the second derivatives (term d2e in the `ebuck2.f` subroutine) were miscalculated in TINKER–VERSION 3.5 when the system is in the repulsive zone. The problem was solved by modifying the `ebuck2.f` subroutine. (If the option Buckingham is selected for the van der Waals energy term, the repulsive wall is estimated by means of a function of R–12; the second derivatives of this function were wrong.)
  - The expression for the first derivatives of the Morse function (term deddt in `ebond1.f` and `ebond2.f`) was miscalculated, in TINKER–VERSION3.5. This was fixed by changing the `ebond1.f` and `ebond2.f` subroutines.
- As explained in Sec. VIII A 1, the Morse treatment for bond lengths has been flexibilized by making it possible to use this option for selected bonds (rather than for all or none of them) and using a user-input bond dissociation energy. The subroutines modified for this purpose were `analyze.f, ebond.f, ebond1.f, ebond2.f, ebond3.f, field.f, kbond.f, prmkey.f, prtprm.f`, and `readprm.f`. The `bndpot.i, bond.i`, and `kbonds.i` include files were also modified.
- As explained in Sec. VIII A 2, when the user selects the Buckingham option for the van der Waals energy term in TINKER 3.5MN1, the energy is calculated by using one expression constructed by means of a linear combination of attractive and repulsive functions. This approach gives results for MM3 calculations (the MM3 force field is the only force field that uses the Buckingham function that is supported in this version of the code) slightly different from those obtained with any other standard code, but continuous with respect to the distance R. The subroutines modified in order to include this approach are `ebuck.f, ebuck1.f, ebuck2.f, ebuck3.f, ebuck4.f`, and `ebuck5.f`.

2. TINKER–*version 3.5mn2*

- Due to a mistake in the treatment of van der Waals 1–4 interactions when using the CHARMM force field present in TINKER–VERSION 3.5, that code or any version of MC–TINKER based on TINKER–VERSION 3.5 should not be used with the CHARMM force field. To prevent incorrect usage, in version 3.5MN2 we removed the CHARMM parameter file `charmm.prm`. We do note that the CHARMM force field is handled correctly in versions 3.7 and later of TINKER (those versions are available from Jay W. Ponder, but they are not compatible with any version of MC-TINKER or TINKERATE).

3. TINKER–*version 3.5mn3*

- This version of the code fixes a bug that caused all charge-dipole interaction energies to have the wrong sign. This involved changes in the following subroutines: `echgdpl.f, echgdpl1.f, echgdpl2.f`, and `echgdpl3.f`.

4. TINKER–*version 3.5mn4*

- A.This version of the code correctly evaluates the van der Waals (VDW) 1–4 interactions.

  It was found that the VDW 1–4 interactions are not correctly calculated in TINKER 3.5 when using the charmm force field. In the charmm force field, a set of specially developed parameters (instead of the normal VDW parameters) is used to evaluate the VDW 1–4 interactions. However, tinker 3.5 does not supply VDW 1–4 parameters in the charmm force field file. Moreover, the data structure in TINKER 3.5 does not provide entries for the VDW 1–4 parameters.

  The data structure in tinker 3.5 was modified to allow the use of the VDW 1–4 parameters. Some subroutines were also changed accordingly. In total, we revised 2 common blocks (`vdw.i` and `kvdws.i`) and 26 subroutines/programs (`readprm.f, prtprm.f, kvdws.f, analyze.f, elj.f, elj1.f, elj2.f, elj3.f, elj4.f, elj5.f, ebuck.f, ebuck1.f, ebuck2.f, ebuck3.f, ebuck4.f, ebuck5.f, egauss.f, egauss1.f, egauss2.f, egauss3.f, ehal.f, ehal1.f, ehal2.f, ehal3.f, ehal4.f`, and `ehal5.f`).

  The charmm force field uses the Lennard-Jones (L-J) function for VDW interactions, but other force fields may use other (e.g. Buckingham) functions for VDW interactions. In both cases, the new VDW 1-4 implementation allows the use of a set of specially developed parameters instead of the normal VDW parameters.

- A new parameter file called `charmm27t35.prm` is also generated, which is based on the `charmm27.prm` file distributed in the TINKER 4.1 package. For proteins and peptides, the charmm27 force field is essentially identical to charmm22 force field. The `charmm27t35.prm` file should be used when the charmm force field is selected. See also Sec. VIII A for discussions on implementations of force field parameters in TINKER.

  The `charmm27t35.prm` was validated by a series of tests. In order to test large systems like the membrane bi-layer, some size-control parameters in the code need modifications. For this propose, five include files were revised: `kangs.i, kbonds.i, ktorsn.i, kiprop.i`, and `kurybr.i`. The larger amount of parameters in `charmm27t35.prm` requires the increase of maximum allowed parameter numbers in `sizes.i`, and the option -QMAXMEM=-1 was used to compile the tinker codes.

  Please note that tinker does not implement the complete set of charmm force field. Thus the charmm27 force field implemented in TINKER 4.1 is not completely the same as the charmm27 force field implemented in original charmm. Users are encouraged to check the tinker manual for more information.

5. TINKER–*version 3.5mn5*

- Subroutines `ebuck1.f, ebuck2.f, ebuck3.f, ebuck4.f` were modified to allow the value for the D coefficient (described in section 5.1.2 of this manual) to be specified in a `.prm` file. See Ref. 4 for the usage of this parameter in MCSI calculations.

- A bug is fixed that is related to improper handling of systems that contain valence bending terms for one MM configuration and no valence bending terms for another MM configuration (e.g., in case of OH + H_2 → H_2O + H reaction). In such cases, MM calculations were carried without valence bending terms in both MM configurations. In fact, this bug was only noticeable when TINKER was used as part of the MCSI code.

  Modified subroutines: `ebuck1.f, ebuck2.f, ebuck3.f, ebuck4.f, energ.f, grads.f, hesss.f`.

B. MCSI

Versions 1.0 through 2009 were called MC-TINKER. Beginning with version 2009-1, the code is called MCSI.

1. MC-TINKER–*version 1.0*

MCSI–version 1.0 is the first official version of MCSI. See Sec. X 9 for discussion of the predecessor of MCSI.

2. MC-TINKER–*version 1.0.1*

MC-TINKER–version 1.0.1 is a bug-fixed version for MC-TINKER–version 1.0.

Updates/corrections are as follows:

- It was found that the initialization was not properly done when tinker subroutines were called for the first time, as compared with calculations done with TINKERATE 8.5. Thus, in version 1.0, the first call gave small errors in the energies and derivatives; sometimes the differences were not negligible. On the

other hand, the rest of the calls to TINKER subroutines were fine. The bug was fixed by adding an additional call to tinker subroutines just for initialization. Since this is a molecular mechanics energy computation, its effect on the CPU time is negligible. The correction affects the following subroutines: `mehook, mghook, mhhook`, and `mctesi`.

- In order to minimize other possible improper initializations, variables are explicitly initialized (e.g., by filling with zero) before calls to TINKER subroutines. The updates affects the following subroutines: `mctesi` and `mcviic`.

- In subroutine `mcviic`, variable tnky was incorrectly defined as double precision. Now it is defined as an integer.

- In subroutines `mccgra (mcches)`, when `e12=0` and `e1=e2`, the gradient (Hessian) was not correctly evaluated. They were obtained as the sum of the two gradients (Hessians). Now they are obtained as the average of the corresponding terms.

- In subroutine `mcptab`, typos in output formats 1400 and 1450 were corrected.

3. MC-TINKER*–version 1.1 (Feburary 2004)*

MC-TINKER–version 1.1 is a new version of MC-TINKER, using TINKER 3.5mn4. The improvements are

- correct evaluations for the VDW 1–4 interactions, and

- implementation of the charmm27 force field. See Sec. XIII A 4 (revision of tinker 3.5mn4) and Sec. X (comments on force fields) for more details.

4. MC-TINKER*–version 1.1.1 (June 2007)*

(Updated by: O.T. and D.G.T.)

MC-TINKER–version 1.1.1 is a bug-fixed version for MC-TINKER–version 1.1.

Updates/corrections are as follows:

- The tensor $\mathbf{C}^{(k)}$ given in eq. (22) of Ref. 3 was not correctly calculated and it was not symmetric. This bug is fixed.

- Because the weight function given in eq. (34) of Ref. 3 is already normalized, the quantity $w(\mathbf{q})$ given in eq. (15) of Ref. 3 is always equal to one. Therefore, eqs. (15), (16), (28), and (29) of Ref. 3 are no longer needed. These equations are not used in the present version.

- There were errors in the previous version of the code that resulted in incorrect MCSI gradients and Hessians. These errors were related to the wrong derivatives of the weight function with respect to the internal coordinates. In particular, the first and second derivatives of the weights given in eq (34) of Ref. 3 with respect to the generalized distances given in eqs. (38) and (40) cannot be correct because these expressions include only the terms that are associated with a distance dk with respect to a single Shepard point k and neglect other distances $dk'(k \neq k')$, while the weight function given in eq. (34) depends on all distances; note the typo in eq. (34): i should be k. Note that when one increments one of the internal coordinates, all distances d change. In the previous version of the code, these derivatives were taken as if one moved a particular Shepard point k rather than changing the current geometry. This resulted in an error in the gradients of the weight, and consequently, in the error in the final MCSI derivatives. In version 1.1.1, we take the derivatives of the weight function given in eq. (34) numerically, because this procedure involves a smaller number of operations, and because it leads to sufficiently accurate final derivatives. Since most of the operations are still analytic, and A.only this one inner derivative in the chain is numerical, the final derivatives may be classified as semi-analytical. Equations (36) through (41) of Ref. 3 are no longer used in the code. The final semi-analytical MCSI first and second derivatives of V are now in excellent agreement with numerical derivatives.

- A typo is corrected in the `mcv12i` subroutine where it calculates off-diagonal elements of a Hessian matrix. That typo resulted in wrong matrix elements of the Hessian of V_{12} when one used different internal coordinates in sets **r** and **s**.

New subroutines added in this version: `mcswtsnum`. Modified subroutines: `mcswts` and `mcv12i`.

5. MC-TINKER*–version 2007 (June 2007)*

(Updated by: O.T. and D.G.T.)

MC-TINKER 2007 is a new version of MC-TINKER, using on TINKER 3.5mn5. The updates are as follows:

The permutation symmetry algorithm is implemented in the MCSI method. To accomplish this, the following subroutines have been added: `mctesis, mcv12is, mehooks, mghooks, mhhooks, mctmmgs, mcrssy, mc12perm, mc23perm, mc13perm, mc123perm, mc132perm, mc12perm2, mc23perm2`, and `mc13perm2`. However, these are added to existing files, so there are no new files. The first five subroutines in this list are counterparts of the corresponding `mctesi`, `mcv12i`, `mehook, mghook`, and `mhhook` subroutines, but they return the symmetrized values.

`mctesis` subroutine returns the Taylor series coefficients D, **b**, and **C** for each symmetrically equivalent Shepard

point (k, i). `mcv12is` subroutine returns the symmetrized coupling term V_{12} and its derivatives in the internal coordinates **r**. `mehooks, mghooks`, and `mhhooks` return the symmetrized MM potential, its gradient and Hessian in Cartesian coordinates (these values are denoted by capital V, **G**, and **F** with a tilde in this manual and in Ref. 4) `mcrssy` subroutine reads in the input in the SYMMETRY section. `mc12perm - mc132perm` subroutines permute rows in a vector. `mc12perm2, mc23perm2`, and `mc13perm2` subroutines permute rows and columns in a square matrix.

The following subroutines have been modified in order to handle permutation symmetry in MCSI calculations: `msptab, mcrfgi, mcrfmm, mcscal, mcsdef, mcsrun, mcv12c`, and `mcviic`.

6. MC-TINKER–*version 2008 (June 2008)*

(Updated by: M.H. and D.G.T.)

MC-TINKER 2008 is a new version of MC-TINKER, using TINKER 3.5mn5. The updates are as follows:

The capability to carry out EE-SCMM and EE-MCSI calculations has been added. Two files, `eemcmm.inc` and `emsrc1.f`, have been added in the srcmct directory. The ISHMM keyword has been added in the RESONANCE section in the `esp.fu85` file. This option specifies whether or not two SCMM Shepard points at which V_{12} and its Taylor series are zero are included for the Shepard interpolation calculation. The GAMESS option of the FORMHESS keyword has been added in the MCGEN83 section in the `esp.fu83` file. Two files, `mcfset.f` and `mcrsmldum.f`, have been added to interface `mc-tinker` 2008 with the sander program in the amber 9. These files are replaced when MC-TINKER 2008 is combined with sander program.

Modified subroutines: `mcptab, mcrfes, mcrfmm, mcrkeh, mcrseg, mcrsep, mcrsre, mcscal, mcsdef, mcsesi, mcsrun`, and `mcswts`.

7. MC-TINKER–*version 2008-2 (June 2008)*

(Updated by: O.T. and D.G.T.)

MC-TINKER 2008-2 is a new version of MC-TINKER, using TINKER 3.5mn5. The updates are as follows:

The u-function given in eq. 19 of Ref. 3 is replaced by another u-function given by eqs. 51 and 97 . This change was necessary to make the global MCSI potential energy surface for OH + HH reaction suitable for semiclassical trajectory calculations. The new u-function is implemented for both symmetrized and nonsymmetrized MCSI calculations. The user can input nondefault values for parameters in the new u-function using the UDELTA and UNN keywords in the MCGENERAL section of the `esp.fu85` file. The new scaling coefficient Y_{ki} (eq. 104), which is based on the value of V_{12ki} at a given geometry, is introduced to calculate the normalized weights w_{ki} (in symmetrized calculations only). This allows one to obtain a good representation of the potential energy surfaces everywhere at large internuclear separation between the reactants, including asymptotic regions. The GRADTEST and the HESSTEST keywords (along with the STEPSIZE keyword) have been added in the MCGENERAL section in the `esp.fu85` file to calculate and compare analytical and numerical MCSI gradients and Hessians. These options will be useful primarily in further developments of the code, but they can also be useful in applications, for example, when one uses new functional forms to represent the molecular mechanics part, or if one wishes to trace a problem (if one encounters a problem) in dynamics calculations.

Modified subroutines: `mcrsge, mcv12i, mcv12is, mcwtss`.

New subroutines: `mcv12isw` returns $F_{ki}(\mathbf{r})$ (eq. 55 of the Appendix). `mcgradtest` calculates MCSI gradient analytically and numerically and prints the results. `mchesstest` calculates MCSI Hessian (half matrix) analytically and numerically and prints the results (both Hessians and their differences for convenience).

8. MC-TINKER–*version 2009 (March 2009)*

(Updated by: O.T. and D.G.T.)

MC-TINKER 2009 is a new version of MC-TINKER, using TINKER 3.5mn5. The updates are as follows:

The non-Hermitian MCSI method and direct Shepard interpolation are implemented. The MCSI code can now carry out calculations using both Hermitian and non-Hermitian MCSI. The keyword NONHERMITIAN is added to the RESONANCE section to indicate the choice of the formalism. The non-Hermitian MCSI scheme is the defaut and is recommended for most cases. Another improvement is the implementation of the Jacobians and Hessians that are necessary for calculation of the first and second derivatives of the weight function with respect to the internal coordinates in the case when set of coordinates **s** is different from set **r**. This is currently only available for a special case of **s** and **r** for a generic atom-transfer reaction as described in Sec. II B 1. The keyword JACOBIANS is added in the RESONANCE section to indicate whether or not to use these Jacobians and Hessians in a calculation.

Subroutines `mcv12i` and `mcv12is` are renamed to `mcv12iold` and `mcv12isold` and they are used to perform Shepard interpolation of modified Taylor series of V_{12} according to the old MCSI method.

The new subroutines: `mcv12i, mcv12is` are added. These subroutines return the interpolated value of β obtained using the non-Hermitian formalism described in Ref. 5.

The subroutines `mccene`, `mccgra`, and `mcches` are modified to carry out calculations according to Hermitian and non-Hermitian MCSI methods.

A new subroutine `mcswtsnum1` is added. This subroutine calculates the first and second derivatives of the weight function in the internal coordinates as described in Sec. II B 1.

9. MCSI–*version 2009-1 (December 2009)*

(Updated by: O.T. and D.G.T.)
MCSI 2009-1 is a new version of MC-TINKER, using TINKER 3.5mn5. The updates are as follows:

In addition to the renaming, the code is updated as follows: A capability is added for constructing potential energy surfaces based only on gradient information, that is without using any Hessian information from electronic structure calculations, as described in Ref. 8. This option is invoked by setting the FORMHESS variable of the `esp.fu83` input file to 'ZERO'.

10. MCSI–*version 2010-1 (October 2010)*

(Updated by: M.H. and D.G.T.)
The updates and corrections are as follows;

- The capability to carry out non-Hermitian EE-MCSI calculations has been added.
- The capability to carry out V_{12} calculation utilizing sines of dihedral angles has been added.
- The ICSHZERO and ICSAZERO keywords in the `esp.fu83` file have been added. The ICEEZERO keyword in the `esp.fu85` file has been added.
- The ZERO option has been added for the FORMDQDPHI and FORMDQDR keywords. The ZERO option of the FORMHESS keyword is now available for EE-MCSI calculations.
- The capability to carry out GRADTEST and HESSTEST options in the TYPECALC keyword for EE-MM and EE-MCSI calculations has been added. The CHARGETEST, DQDRTEST, and DQDPHITEST options in TYPECALC keyword have been added.
- In non-Hermitian (EE-)MCSI calculations, not $[V_{12}]^2$ but V_{12} is now printed to match the units of energy and diagonal matrix elements.
- A bug in calculating the EE-MM energy has been fixed.
- Some comment lines starting with 'CAMBP' has been added in the program. They are activated in compiling the AMBERPLUS program.

New subroutines: `mcishz, mcv12z, eechargetest, eedqdphitest, eedqdrtest, eestiz, eev12iold`.
Modified subroutines: `main, mcrkic, mcprez, mcrfgi, mcrkeh, mcrsep, mcrsge, mcsdef, mcsisi, mcgradtest, mchesstest, mcswts, eerd85, mcv12iold, mcv12i1st, mcv12i, eetiqr, eev12i, eeviic`

---

1 O. Tishchenko, M. Higashi, T. V. Albu, J. C. Corchado, Y. Kim, J. Vill, J. Xing, H. Lin, and D. G. Truhlar, MCSI–version 2010-1 University of Minnesota, Minneapolis, MN, 2010.
2 J. W. Ponder, TINKER–version 3.5, Washington University, St. Louis, MO, 1997.
3 Y. Kim, J. C. Corchado, J. Villa, X. Xing, and D. G. Truhlar, J. Chem. Phys. **112**, 2718 (2000).
4 O. Tishchenko and D. G. Truhlar, J. Chem. Theory Comp. **3**, 938 (2007).
5 O. Tishchenko and D. G. Truhlar, J. Chem. Theory Comp. **5**, 1454 (2009).
6 M. Higashi and D. G. Truhlar, J. Chem. Theory Comp. **4**, 790 (2008).
7 O. Tishchenko and D. G. Truhlar, J. Chem. Phys. **130**, 024105 (2009).
8 O. Tishchenko and D. G. Truhlar, J. Chem. Phys. **132**, 084109 (2010).
9 F. London, Z. Electrochem. **35**, 552 (1929).
10 H. Eyring and M. Polanyi, Z. Phys. Chem. **B12**, 279 (1931).
11 G. E. Kimball and H. Eyring, J. Am. Chem. Soc. **54**, 3876 (1932).
12 C. A. Coulson and U. Danielsson, Arkiv Fysik **8**, 239 (1954).
13 S. Sato, Bull. Chem. Soc. Japan **28**, 450 (1955).
14 F. O. Ellison, J. Am. Chem. Soc. **85**, 3540 (1963).
15 J. K. Cashion and D. R. Herschbach J. Chem. Phys. **40**, 2358 (1964).
16 C. A. Parr and D. G. Truhlar, J. Phys. Chem. **75**, 1844 (1971).
17 L. M. Raff, J. Chem. Phys. **40**, 2358 (1964).
18 P. J. Kuntz, In Atom-Molecule Collision Theory; Bernstein, R. B., Ed.; Plenum: New York, 1979; p. 79.
19 A. Warshel and R. M. Weiss, J. Am. Chem. Soc. **102**, 6218 (1980).
20 A. Warshel, S. Russell, and R. M. Weiss, In Chemical Approaches to Understanding Enzyme Catalysis: Biomimectic Chemistry and Transition State Analogs; B. S. Green, Y. Ashani, and D. Chipman, Eds.; Elsevier: Amsterdam, 1981, p. 267.
21 A. Warshel, Computer Modeling of Chemical Reactions in Enzymes and Solutions; John Wiley & Sons: New York, 1991.
22 Y. T. Chang and W. H. Miller, J. Phys. Chem. **94**, 5884 (1990).
23 Y. T. Chang, C. Minichino, and W. H. Miller, J. Chem. Phys. **96**, 4341 (1992).
24 P. Pulay and G. Fogarasi, J. Chem. Phys. **96**, 2856 (1992).
25 Y.-Y. Chuang and D. G. Truhlar, J. Phys. Chem. A **102**, 242 (1998).
26 J. Ischtwan and M. A. Collins, J. Chem. Phys. **100**, 8080 (1994).
27 K. A. Nguyen, I. Rossi, and D. G. Truhlar, J. Chem. Phys. **103**, 5522 (1995).
28 T. V. Albu, J. C. Corchado, and D. G. Truhlar, J. Phys. Chem. A **105**, 8465 (2001).
29 H. Lin, J. Pu, T. V. Albu, and D. G. Truhlar, J. Phys. Chem. A **108**, 4112 (2004).
30 H. Lin, Y. Zhao, O. Tishchenko, and D. G. Truhlar, J. Chem. Theory Comp. **2**, 1237 (2006).
31 O. Tishchenko and D. G. Truhlar, J. Phys. Chem. A **110**,

13530 (2006).
32 J. W. Downing and J. Michl, In Potential Energy Surfaces and Dynamics Calculations; D. G. Truhlar,, Ed.; Plenum: New York, 1981; p. 199.
33 A. Morita and S. Kato, J. Am. Chem. Soc. **119**, 4021 (1997).
34 S. Iuchi, A. Morita, and S. Kato, J. Phys. Chem. B **106**, 3466 (2002).
35 Z. Lu and W. Yang, J. Chem. Phys. **121**, 89 (2004).
36 M. Higashi and D. G. Truhlar, J. Chem. Theory Comp. **5**, 2925 (2009).