2007

Direct Dynamics Simulations Using Hessian-Based Predictor-Corrector Integration Algorithms

Upakarasamy Lourderaj  
*Texas Tech University*

Kihyung Song  
*Korea National University of Education*

Theresa Lynn Windus  
*Iowa State University*, twindus@iastate.edu

Yu Zhuang  
*Texas Tech University*

William L. Hase  
*Texas Tech University*

Follow this and additional works at: [http://lib.dr.iastate.edu/chem_pubs](http://lib.dr.iastate.edu/chem_pubs)  
Part of the [Chemistry Commons](http://lib.dr.iastate.edu/chem_pubs)

The complete bibliographic information for this item can be found at [http://lib.dr.iastate.edu/chem_pubs/928](http://lib.dr.iastate.edu/chem_pubs/928). For information on how to cite this item, please visit [http://lib.dr.iastate.edu/howtocite.html](http://lib.dr.iastate.edu/howtocite.html).
Direct Dynamics Simulations Using Hessian-Based Predictor-Corrector Integration Algorithms

Abstract
In previous research [J. Chem. Phys.111, 3800 (1999)] a Hessian-based integration algorithm was derived for performing direct dynamics simulations. In the work presented here, improvements to this algorithm are described. The algorithm has a predictor step based on a local second-order Taylor expansion of the potential in Cartesian coordinates, within a trust radius, and a fifth-order correction to this predicted trajectory. The current algorithm determines the predicted trajectory in Cartesian coordinates, instead of the instantaneous normal mode coordinates used previously, to ensure angular momentum conservation. For the previous algorithm the corrected step was evaluated in rotated Cartesian coordinates. Since the local potential expanded in Cartesian coordinates is not invariant to rotation, the constants of motion are not necessarily conserved during the corrector step. An approximate correction to this shortcoming was made by projecting translation and rotation out of the rotated coordinates. For the current algorithm unrotated Cartesian coordinates are used for the corrected step to assure the constants of motion are conserved. An algorithm is proposed for updating the trust radius to enhance the accuracy and efficiency of the numerical integration. This modified Hessian-based integration algorithm, with its new components, has been implemented into the VENUS/NWChem software package and compared with the velocity-Verlet algorithm for the H2CO→H2+CO, O3+C3H6, and F−+CH3OOH chemical reactions.

Keywords
Interpolation, Ab initio calculations, Trajectory models, Electronic structure, Angular momentum

Disciplines
Chemistry

Comments
The following article appeared in Journal of Chemical Physics 126 (2007): 044105, and may be found at doi:10.1063/1.2437214.

Rights
Copyright 2007 American Institute of Physics. This article may be downlo (aded for personal use only. Any other use requires prior permission of the author and the American Institute of Physics.
Direct dynamics simulations using Hessian-based predictor-corrector integration algorithms

Upakarasamy Lourderaj, Kihyung Song, Theresa L. Windus, Yu Zhuang, and William L. Hase

Citation: The Journal of Chemical Physics 126, 044105 (2007); doi: 10.1063/1.2437214
View online: http://dx.doi.org/10.1063/1.2437214
View Table of Contents: http://scitation.aip.org/content/aip/journal/jcp/126/4?ver=pdfcov
Published by the AIP Publishing

Articles you may be interested in

- Direct dynamics simulation of dioxetane formation and decomposition via the singlet ·O–O–CH2–CH2· biradical: Non-RRKM dynamics

- Theoretical investigation of the dissociation dynamics of vibrationally excited vinyl bromide on an ab initio potential-energy surface obtained using modified novelty sampling and feedforward neural networks. II. Numerical application of the method

- Theoretical investigation of the dissociation dynamics of vibrationally excited vinyl bromide on an ab initio potential-energy surface obtained using modified novelty sampling and feed-forward neural networks

- Accurate reaction paths using a Hessian based predictor–corrector integrator
  J. Chem. Phys. 120, 9918 (2004); 10.1063/1.1724823

- Ab initio classical trajectories on the Born–Oppenheimer surface: Hessian-based integrators using fifth-order polynomial and rational function fits
Direct dynamics simulations using Hessian-based predictor-corrector integration algorithms

Upakarasamy Lourderaj
Department of Chemistry and Biochemistry, Texas Tech University, Lubbock, Texas 79409

Kihyung Song
Department of Chemistry, Korea National University of Education, Chongwon, Chungbuk 363-791, Korea

Theresa L. Windus
Department of Chemistry, Iowa State University, Ames, Iowa 50010

Yu Zhuang
Department of Computer Science, Texas Tech University, Lubbock, Texas 79409

William L. Hase
Department of Chemistry and Biochemistry, Texas Tech University, Lubbock, Texas 79409

(Received 1 November 2006; accepted 4 January 2007; published online 29 January 2007)

In previous research [J. Chem. Phys. 111, 3800 (1999)] a Hessian-based integration algorithm was derived for performing direct dynamics simulations. In the work presented here, improvements to this algorithm are described. The algorithm has a predictor step based on a local second-order Taylor expansion of the potential in Cartesian coordinates, within a trust radius, and a fifth-order correction to this predicted trajectory. The current algorithm determines the predicted trajectory in Cartesian coordinates, instead of the instantaneous normal mode coordinates used previously, to ensure angular momentum conservation. For the previous algorithm the corrected step was evaluated in rotated Cartesian coordinates. Since the local potential expanded in Cartesian coordinates is not invariant to rotation, the constants of motion are not necessarily conserved during the corrector step. An approximate correction to this shortcoming was made by projecting translation and rotation out of the rotated coordinates. For the current algorithm unrotated Cartesian coordinates are used for the corrected step to assure the constants of motion are conserved. An algorithm is proposed for updating the trust radius to enhance the accuracy and efficiency of the numerical integration. This modified Hessian-based integration algorithm, with its new components, has been implemented into the VENUS/NWChem software package and compared with the velocity-Verlet algorithm for the H$_2$CO→H$_2$+CO, O$_3$+C$_3$H$_6$, and F$^-$+CH$_3$OOH chemical reactions. © 2007 American Institute of Physics. [DOI: 10.1063/1.2437214]

I. INTRODUCTION

Classical trajectory chemical dynamics simulations, where the classical equations of motion of the atoms are numerically integrated on a potential energy surface (PES), have been used to study chemical reactions since the early 1960s. For the most part, the simulations are performed solving Hamilton’s equations of motion with the gradient obtained from the PES. Well-established integration schemes are used for the calculations. In the traditional approach, the surface is represented by an analytic function obtained by fitting ab initio and/or experimental data. Exact fitting of the ab initio data is feasible for systems with a small number of atoms or a high degree of symmetry. For large systems, the number of internal degrees of freedom to which the data are fitted becomes large (3N-6 unique degrees of freedom for a nonlinear system of N atoms with no symmetry). In such cases, often a model analytic potential is derived for a few degrees of freedom that are important to describe, the reaction and empirical potentials are used for the other degrees of freedom. However, identifying the important degrees of freedom has considerable uncertainty.

In recent work, additional approaches and algorithms have been proposed and used for representing PESs. If the potential energy and gradient are available at each point of the numerical integration from an electronic structure theory calculation, the trajectories may be integrated “on the fly,” as first demonstrated by Wang and Karplus. In such a “direct dynamics” simulation, the local potential and gradient are determined directly from an electronic structure theory during the numerical integration. With the increase in speed of computers and improvements in electronic structure algorithms, a large number of reactions have been studied using direct dynamics. However, calculation of the potential and gradient for each integration step of a trajectory becomes computationally very expensive for large molecular systems and/or high levels of electronic structure theory.

One strategy for applying direct dynamics to large systems is to modify a semiempirical electronic structure theory by refitting some or all of its parameters to accurate experi-
mental data and/or high-level \textit{ab initio} calculations for a specific reaction.\textsuperscript{20–26} These new parameters are called specific reaction parameters (SRPs).\textsuperscript{20} To calculate a trajectory the potential energy and gradient are obtained directly from this semiempirical-SRP electronic structure theory. For some systems this approach has led to reliable potential energy surfaces,\textsuperscript{22,24–26} but for others the semiempirical model used was not sufficiently flexible to accurately represent experiment and high-level \textit{ab initio} calculations.\textsuperscript{21,23} Single reference and minimal basis set semiempirical models, such as AM1 and PM3, may not incorporate sufficient detail of the quantum mechanics to accurately represent the PES. In using these models to study H$_2$CO $\rightarrow$ H$_2$+CO product energy partitioning, it was found that the results of the trajectory simulation depended on the specific information used to fit the SRPs.\textsuperscript{21} A semiempirical-SRP electronic structure theory model which appears to be substantially more accurate and broadly applicable is one based on a floating-occupation molecular orbital configuration interaction wave function.\textsuperscript{25–27}

A number of different algorithms have been presented for developing analytic PESs by interpolating \textit{ab initio} data points.\textsuperscript{28–37} Ischtwan and Collins have proposed\textsuperscript{28} and used extensively\textsuperscript{29} a Shepherd interpolation scheme. In recent work\textsuperscript{30} this method has been modified by using Bayesian analysis to define confidence regions for the interpolation. Ho and Rabitz\textsuperscript{31} proposed an interpolation procedure within the framework of a reproducing kernel Hilbert space, and this method has been used by Schatz and co-workers\textsuperscript{32,33} in chemical dynamics simulations. Salazar and co-workers\textsuperscript{34,35} have proposed a method similar to that of Collins and co-workers,\textsuperscript{28–30} but different in the definition of the local fitting region and the interpolation. Eckert and Werner\textsuperscript{36} have proposed a least-square interpolation scheme, related to the method of Collins and co-workers, but different in that it retains strict locality in a rectangular grid. Maisuradze \textit{et al.}\textsuperscript{37} have presented an interpolating moving least-squares method, which removes a possible discontinuity in the method of Ischtwan and Collins.

There is considerable interest in determining the chemical dynamics, whether classical, semiclassical, or quantum, predicted by a specific electronic structure theory. In the same way, as one has studied how stationary point and reaction path properties depend on the level of the electronic structure theory, one can investigate the relationship between the predicted chemical dynamics and the electronic structure theory method. If the interpolation is converged, when using the above schemes for interpolating electronic structure theory data points, the exact chemical dynamics for the theoretical method will be obtained. However, a very fine interpolation may be required to obtain the exact trajectory. Another approach is direct dynamics described above, in which the trajectory is integrated using direct information from the electronic structure theory. To ensure the accuracy of the simulations, the need for time-reversible trajectories\textsuperscript{38,39} and a strict self-consistent-field (SCF) convergence criterion has been stressed.\textsuperscript{40}

These direct dynamics simulations become computationally quite expensive for a high-level electronic structure theory and it is, thus, important to use the largest numerical integration step size while maintaining the accuracy of the trajectory. To use a larger integration step, Helgaker \textit{et al.}\textsuperscript{41} proposed a scheme that uses the second derivative of the potential. If the second derivatives of the potential (Hessians) are given directly by the electronic structure theory, a local approximation to the true PES can be made using a second-order Taylor expansion and the trajectories can be calculated using this approximate potential. The local quadratic potential is only valid in a small region, called a “trust region” defined by a trust radius. The equations of motion are integrated to the end of the trust radius, where the potential, gradient, and Hessian are calculated again. The new potential, gradient, and Hessian define a new local quadratic PES on which the integration of the equations of motion is continued. Since the potential, gradients, and Hessians are known at the starting and the ending of each integration step, Millam \textit{et al.}\textsuperscript{42} used a fifth-order polynomial or a rational function to fit the potential between the two points and correct the trajectory. This gives a more accurate trajectory in the trust region and allows one to take larger integration steps. The integration on the approximate quadratic model potential is called the “predictor step” and that on the fifth-order PES is called the “corrector step.” The computational cost for the Hessian evaluations can be considerably reduced if some are approximated using a Hessian updating scheme. Bakken \textit{et al.}\textsuperscript{43} tested different Hessian updating schemes and found Bofill’s updating scheme\textsuperscript{44} most suitable for Hessian-based integrators.

In the study reported here, improvements to the Hessian-based integration scheme of Millam \textit{et al.}\textsuperscript{42} were developed. A predictor step using the quadratic potential model is taken to the end of the trust radius using the velocity-Verlet integration algorithm\textsuperscript{45} to integrate Newton’s equation, in Cartesian coordinates. At the end of the predictor step, the potential, gradients, and Hessian are calculated again. The exact potential, gradient, and Hessian at the beginning and the end of the predictor step are used to interpolate a fifth-order potential between the two points as proposed by Millam \textit{et al.}\textsuperscript{42} However, new schemes are derived for interpolation of the potential and gradients. For one, the predictor interpolation and integration are done in Cartesian space. This ensures that the linear momentum always remains conjugate to the coordinates and hence angular momentum is well conserved. The trust radius for the next step is updated using a new algorithm based on the difference between the predicted and the corrected trajectories. The Hessians are updated using Bofill’s scheme,\textsuperscript{44} as described by Bakken \textit{et al.}\textsuperscript{43}

Millam \textit{et al.} performed the predictor integration step in instantaneous normal mode coordinates,\textsuperscript{46} an approach which does not conserve angular momentum even if the initial total angular momentum is zero.\textsuperscript{47,48} In addition, their corrector interpolation and integration were performed in a rotated Cartesian space, which does not conserve angular momentum,\textsuperscript{49} since the local potential expanded in Cartesian coordinates is not rotationally invariant.\textsuperscript{50} Angular momentum was conserved by a subsequent numerical treatment of the predicted trajectory.\textsuperscript{44} For the Hessian-based model described here, Cartesian coordinates are used for integration of Newton’s equations of motion for both the predictor and
corrector steps. If the interpolation of the fifth-order predicted potential is done in the rotated coordinates, angular momentum is preserved by rotating the coordinates and gradients back to the original Cartesian space, in which the numerical integration of the equations of motion is performed.

In the following section, the methodology of the proposed Hessian-based integration scheme is described. Section III then presents tests of the integration scheme for 

II. METHODOLOGY

A. Predictor step: Second-order potential energy

In gradient-based Born-Oppenheimer direct dynamics, the potential energy $V(q)$ and gradient $\partial V/\partial q_i$ are evaluated at each step during the integration of the trajectory. At any point $q_1$ during the integration, the potential around $q_1$ can be expanded using a quadratic model,

$$V(q) = V_1 + G_1^T \Delta q + \frac{1}{2} \Delta q^T H_1 \Delta q,$$

where $\Delta q = q - q_1$, and $V_1$, $G_1$, and $H_1$ are the energy, gradient, and Hessian evaluated at $q_1$. Hence, in a region for which the quadratic approximation is sufficiently accurate, the above quadratic potential can be used to integrate the equations of motion, without the evaluation of the potential energy and gradients at each integration step. With this quadratic potential, the Newton’s equations of motion in Cartesian coordinates become

$$m\ddot{q} = -G_1 - H_1 \Delta q,$$

and they may be integrated using standard numerical procedures. The local quadratic potential is valid only in a small region, defined by a “trust radius,” $R = (\sum_{i=1}^{3N} \Delta q_i^2)^{1/2}$. At the end of the trust radius $q_2$, the energy ($V_2$), gradient ($G_2$), and Hessian ($H_2$) are calculated. This completes the predictor step of the Hessian-based integrator which is based on the integration schemes used in Refs. 41 and 42. As described below, a new method is used to adjust the trust radius based on the accuracy of the current step. The trust radius is fixed in Ref. 42. Also, in our algorithm the velocity-Verlet routine is used to integrate the classical equations of motion, i.e., Eq. (2). As discussed in the Introduction, this approach assures conservation of angular momentum. In Ref. 42 instantaneous normal modes, which do not include angular momentum, were used to solve the classical equations.

B. Corrector step: Fifth-order potential energy

After integrating from $q_1$ to the end of the trust radius $q_2$, one has the energy, gradients, and Hessians at the two points. Using this information, this region of the potential energy surface may be approximated by a fifth-order polynomial. The classical equations of motion can be integrated again using standard procedures, on the fifth-order polynomial surface. The trajectory in this corrector step represented by $q(t)$ needs the potential and the gradient for the numerical integration. It is convenient to decompose the coordinates to parallel ($q_3$), and perpendicular ($q_4$) components to perform the interpolation.42 The parallel component lies parallel to the vector $(q_2 - q_1)$, and the remaining coordinates form the perpendicular components. The potential is accurate to fifth order only along the parallel component, and to second order in other directions. Two hyperplanes $P_1$ and $P_2$ that contain $q_1$ and $q_2$, respectively, are defined such that $(q_2 - q_1)$ is perpendicular to both the planes $P_1$ and $P_2$ (Fig. 1). Interpolation of the potential and the gradient at $q(t)$ is performed by first projecting $q(t)$ onto the two hyperplanes $P_1$ and $P_2$, as shown in Fig. 1.

The interpolation of the local potential may be done in either a rotated coordinate system or in general Cartesian coordinates. In the rotated coordinates, $q_1(q_1^1, q_1^2, q_1^3, \ldots, q_1^{3N})$ and $q_2(q_2^1, q_2^2, q_2^3, \ldots, q_2^{3N})$ are first translated such that $q_1$ is the origin. For simplicity, the same notations $q_1$ and $q_2$ are used to represent the two points in the translated and rotated coordinates. The coordinates are rotated such that the vector $q_2$ lies along one of the coordinate axes. In the present case, the $x$ axis of atom 1 is chosen; i.e., $q_1^1$. This defines the unitary transformation matrix $U$, so that $q_2$ is rotated according to

$$\begin{pmatrix} q_1^1 \\ 0 \\ \vdots \end{pmatrix} = U \begin{pmatrix} q_2^1 \\ q_2^2 \\ q_2^3 \\ \vdots \end{pmatrix},$$

where the unitary matrix $U$ written in terms of $q_2$ is
nate rotation is given by

$$Q_{\text{rot}} = U^T Q,$$

(5)

where $Q$ is the original Cartesian coordinate vector and $Q_{\text{rot}}$ is the rotated coordinate vector. Translation and rotation of the coordinates are illustrated in Fig. 2. The gradients and Hessians used in the interpolation scheme also need to be rotated by

$$g = U^T G, \quad h = U^T H U,$$

(6)

where $g$ and $h$ are the rotated gradients and Hessians.

To perform the interpolation during a corrector integration step, the coordinate vector $q(t)$ is first translated and rotated using Eq. (3) and then projected perpendicularly onto the two hyperplanes $P_1$ and $P_2$. Let the projected points on $P_1$ and $P_2$ be $q_4$ and $q_5$, respectively (Fig. 1). The projected coordinates $q_4$ and $q_5$ differ only in their parallel components. The potentials at $q_4$ and $q_5$ are given by

$$V_4(q_4) = V_1 + g_1^T \Delta q_1 + \frac{1}{2} \Delta q_1^T h_1 \Delta q_1,$$

(7)

$$V_5(q_5) = V_2 + g_2^T \Delta q_2 + \frac{1}{2} \Delta q_2^T h_2 \Delta q_2.$$  

(8)

The potential at $q(t)$ can be written in terms of the parallel and perpendicular components as

$$V(q) = V(q_{\perp}, s) = V(q_4 + s(q_5 - q_4))$$

$$= V(q_{\perp} + s(q_5 - q_4) + q_4),$$

(9)

where $s = (q_5 - q_4)/|q_5 - q_4|$; i.e., when $s = 0$ it gives the potential at $q_4$ and at $s = 1$ it gives the potential at $q_5$. The potential and its first and second derivatives with respect to $s$ at $s = 0$ and $s = 1$ are given by

$$V(q_{\perp}, s)|_{s=0} = V_4, \quad V(q_{\perp}, s)|_{s=1} = V_5,$$

$$\frac{\partial V(q_{\perp}, s)}{\partial s} |_{s=0} = (g_1 + h_1 \Delta q_1)(q_2 - q_1),$$

$$\frac{\partial V(q_{\perp}, s)}{\partial s} |_{s=1} = (g_2 + h_2 \Delta q_2)(q_2 - q_1),$$

$$\frac{\partial^2 V(q_{\perp}, s)}{\partial s^2} |_{s=0} = (q_2 - q_1)^T h_1 (q_2 - q_1).$$

(10)

$$\frac{\partial^2 V(q_{\perp}, s)}{\partial s^2} |_{s=1} = (q_2 - q_1)^T h_2 (q_2 - q_1).$$

(11)

and $r_n = (\sum_{i=3N}^{3N} q_i^2)^{1/2}$; $n = 1$ to $3N$. The complete coordinate rotation is given by

$$U = \begin{pmatrix}
q_1^2/r_{n-1} & q_2^2/r_{n-1} & \cdots & q_{3N}^2/r_{n-1} \\
- r_{n-2}/r_{n-1} & q_2^2/r_{n-2} & q_1^2 & \cdots & q_{3N}^2/r_{n-2} \\
0 & - r_{n-3}/r_{n-2} & q_2^2/r_{n-3} & \cdots & q_{3N}^2/r_{n-3} \\
0 & 0 & - r_{n-4}/r_{n-3} & \cdots & q_{3N}^2/r_{n-4} \\
0 & 0 & 0 & \cdots & - q_{3N}/r_1
\end{pmatrix},$$

(4)

$$\frac{\partial^2 V(q_{\perp}, s)}{\partial s^2} |_{s=1} = (q_2 - q_1)^T h_2 (q_2 - q_1).$$

Since six pieces of information are available, the potential $V(q_{\perp}, s)$ can be expanded as a fifth-order polynomial in $s$,

$$V(q_{\perp}, s) = a_1 + a_2 s + a_3 s^2 + a_4 s^3 + a_5 s^4 + a_6 s^5.$$  

(11)

Evaluating the potential and first and second derivatives of Eq. (11) for $s = 0$ and $s = 1$ and equating them to Eq. (10), the coefficients of the polynomial in Eq. (11) can be obtained by solving the equation, $\lambda a(q_{\perp}) = e(q_{\perp})$, i.e.,

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 2 & 6 & 12 & 20 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} = \begin{pmatrix} V(q_{\perp}, 0) \\ \partial V(q_{\perp}, 0)/\partial s \\ \partial^2 V(q_{\perp}, 0)/\partial s^2 \\ V(q_{\perp}, 1) \\ \partial V(q_{\perp}, 1)/\partial s \\ \partial^2 V(q_{\perp}, 1)/\partial s^2 \end{pmatrix}.$$  

(12)

The gradient at $q$ is given by

$$\frac{\partial V}{\partial q} = \frac{\partial V(q_{\perp}, s)}{\partial s} \frac{(q_2 - q_1) (q_2 - q_1)}{|q_2 - q_1|^2} + \frac{\partial^2 V(q_{\perp}, s)}{\partial q_{\perp}^2}.$$  

In the rotated coordinates, the projection of $q(t)$ onto the hyperplanes $P_1$ and $P_2$ is simplified, as the projected points differ only in their parallel components. However, care must be taken to rotate the coordinates and gradients back to the Cartesian frame, for each integration step, to remove any spurious angular momentum generated due to the rotation of the coordinates.

As discussed above, the interpolation of the potential may also be performed in Cartesian coordinates, without transforming to the rotated coordinates. If the coordinates are not translated and rotated as described above, the projection of $q$ perpendicular to the hyperplanes $P_1$ and $P_2$, at $q_4$ and $q_5$ is performed as follows. Since $(q_4 - q)$ is parallel to $(q_2 - q_1)$,
\[(q_4 - q) = c_1(q_2 - q_1), \]  

(14a) 

so that 
\[
\frac{(q_4 - q^0)}{(q_2 - q^0)} = \frac{(q_2^0 - q_1^0)}{(q_2^0 - q_1^0)} = c_1  
\] 

(14b) 

for some coefficient \(c_1\). The coefficient \(c_1\) can be found by equating Eq. (14b) to a known component of \((q_2^0 - q_1^0)\) represented as \((q_2^0 - q_1^0)\), where the index 0 may be any one of \(1, 2, \ldots, 3M\), as long as \((q_2^0 - q_1^0) \neq 0\). Using the maximum of \((q_2^0 - q_1^0)\) is a good choice. The projected coordinate \(q'_4\) may now be written as 
\[
q'_4 = \frac{(q_4^0 - q_1^0)}{(q_2^0 - q_1^0)}(q_0^0 - q_1^0) + q_1. \]  

(15) 

Here, all the variables are known except \(q_4^0\), which may be found by using the perpendicular property of the vectors \((q_2 - q_1)\) and \((q_4 - q_1)\). Since the planes \(P_1\) and \(P_2\) are perpendicular to \((q_2 - q_1)\), and \(q_1\) and \(q_4\) lie on \(P_1\), 
\[
(q_4 - q_1) \cdot (q_2 - q_1) = 0, \]  

(16a) 

so that 
\[
\sum_{i=1}^{3N} (q_2^0 - q_1^0)q_i^0 = \sum_{i=1}^{3N} (q_2^0 - q_1^0)q_i^0, \]  

(16b) 

where “\(\cdot\)” denotes the inner product of the vectors. Substituting Eq. (15) into Eq. (16b) gives 
\[
q_4^0 = \frac{\sum_{i=1}^{3N}[(q_2^0 - q_1^0)(q_2^0 - q_1^0)q_i^0 + q_1^0]}{\sum_{i=1}^{3N}[(q_2^0 - q_1^0)^2]}. \]  

(17) 

Similar equations may also be obtained for \(q_5\). The potential and the gradient at \(q(t)\) can be obtained in a similar way as described by Eqs. (7)–(13), with \(s = (q - q_4)(q_2 - q_1)/||q_2 - q_1||^2\). However, the gradient must be projected perpendicularly onto the hyperplane passing through the origin. The equations of motion for this corrector step are reintegrated for the same time interval, with the velocity-Verlet algorithm, as in the predictor step.

### C. Trust radius update

The accuracy of the above Hessian-based integration scheme depends on the trust radius. If the trust radius is too large, the second-order expansion of the potential is not sufficiently accurate, resulting in a trajectory that deviates substantially from the true one. Thus, the corrector step will also fail. An accurate trajectory may be followed by using a small trust radius. However, a very small trust radius results in more ab initio calls, which unnecessarily slows down the algorithm. Thus, care must be taken to update the trust radius to allow the largest possible predictor step, without deviating far away from the true trajectory. Choosing such a trust radius is a very important component of the algorithm.

The general method of updating the trust radius is given by
\[
R_{t+1} = R_t F, \]  

(18) 

where \(R_{t+1}\) is the trust radius of the next step, \(R_t\) the trust radius of the current step, and \(F\) a scaling factor. In optimization algorithms, where the concept of a trust radius is used, the scaling factor \(F\) is based on the accuracy of the potential in the current trust radius \((R_t)\).44,52 In Refs. 44 and 52, the estimate of the accuracy of the current trust radius step is written as the ratio, \(r = (V_l - V_i)/(V_m - V_i)\), where \(V_m\) is the quadratic potential at \(q_2\) and \(V_i\) and \(V_l\) are the actual potentials at \(q_1\) and \(q_2\). Updating of the trust radius is done based on the value of \(r\). If \(r_{\text{min}} < r < r_{\text{max}}\), then the trust radius is increased, otherwise, the trust radius is decreased. The parameters \(r_{\text{min}}\) and \(r_{\text{max}}\) are arbitrary values, which are close to 1. A scaling factor, \(F\), of \(\sqrt{2}\) is used commonly for increasing the trust radius and 1/2 for decreasing the trust radius.44,52

The major issue of the above scheme is its inflexibility, in that the trust radius is only adjusted by a constant scaling factor depending on the error tolerance.

In the present study, a new trust radius updating scheme suitable for trajectory simulations was developed. Since the accuracy of the simulation is related to the fact that the predictor and corrector trajectories should be close to each other, this new updating scheme is based on the difference between the predictor and corrector trajectories.4 The perpendicular component of the distance between the ends of the predictor and corrector steps \(\Delta q_4 = ||q_4 - q_3||\) was compared to a threshold value \(r_0\). The trust radius was chosen based on how far \(\Delta q_4\) is away from \(r_0\) which defines the error in the trajectory, \(r = \Delta q_4 / r_0\). The scaling factor, \(F\), in Eq. (18), is written as a function of the error, \(\rho\). Here the Gaussian function
\[
F(\rho) = \alpha_1 + \alpha_2 e^{-\rho^2/\beta}, \]  

(19) 

is used as the scaling function, where \(\alpha_1, \alpha_2, \) and \(\beta\) are parameters. The use of a Gaussian function as the scaling function allows flexibility in the variation of the trust radius in contrast to the scheme discussed above. The parameters \(\alpha_1\) and \(\alpha_2\) define the range that the scaling function can take which are chosen by physical significance. The parameter \(\alpha_1\) defines the lower limit of the scaling function and is chosen such that \(\alpha_1 > 0\) to avoid \(F(\rho)\) becoming zero for large values of \(\rho\). Good choices for the parameters are \(\alpha_1 = 0.5\) and \(\alpha_2 = 1.0\), which let the scaling function vary between 0.5 and 1.5. The parameter \(\beta\) is found by using the condition that, for \(\rho = 1\), the error in the correct trajectory equals the tolerence and the trust radius is not scaled; i.e., \(F(\rho) = 1\) for \(\rho = 1\). Thus, the parameter \(\beta\) is written as
\[
\alpha_1 + \alpha_2 e^{-1/\beta} = 1, \]  

(20a) 

\[-\frac{1}{\beta} = \ln \left(1 - \frac{\alpha_1}{\alpha_2}\right). \]  

(20b) 

For \(\alpha_1 = 0.5\) and \(\alpha_2 = 1.0\), \(\beta\) is 1.44.

In tests of this updating algorithm, i.e., Eqs. (18)–(20), it was found that, if a large trust radius resulted in an accurate trajectory, at times the even larger trust radius given by the updating resulted in a highly inaccurate integration step. Thus, the quadratic approximation to the potential was not
valid within this increased trust radius. Thus, in the present implementation, \( R_t \) in Eq. (18) is replaced by \( R_e \), a constant, so that the trust radius fluctuates around a constant value. A value for \( R_e \) is chosen by calculating several initial trajectories with different values for this parameter.

The algorithm used for the trust radius is:

(i) Define

\[ \rho = \| q_{1k} - q_{2k} \| / r_0, \]

(ii) and then the value of the next trust radius is

\[ R_{t,k+1} = R_e F(\rho). \]  

(21)

\( F(\rho) \) is a Gaussian scaling function, given by Eq. (19). A parabolic function can also be used for this.

The current approach allows a small trust radius in sensitive regions of the PES and an increase in the trust region in insensitive regions. The former maintains accuracy for the trajectory, while the latter maintains accuracy and shortens the computation time. For large \( \rho \), \( F(\rho) \) equals \( \rho \) and increases to \( \alpha_1 + \alpha_2 \) for \( \rho = 0 \). \( F(\rho) \) is restricted to take the lower limit of \( \alpha_1 \) to avoid severe penalizing of the numerical integration by allowing the trust radius \( R_{t,k+1} \) to become quite small. Testing different trust radius updating schemes and identifying their stabilities will be explored in detail in a subsequent paper.

D. Hessian update

While the use of the Hessians at two points allows one to take larger integration steps, a substantial speedup can be achieved, if the Hessians can be approximated instead of evaluating them at each step. The updating scheme is shown here for completeness of the complete integration algorithm. Bofill’s Hessian updating scheme \(^{54} \) for transition state optimization was used for this purpose. Bofill’s scheme is a combination of Murtagh and Sargent \(^{53} \) (MS) and Powell \(^{54} \) (P) updates,

\[ H_{\text{Bofill}}^{k+1} = (1 - \phi_k) H_{\text{MS}}^{k+1} + \phi_k H_{\text{P}}^{k+1}, \]

where

\[ H_{\text{MS}}^{k+1} = H^0 + \frac{(\Delta g - H^0 \Delta q)(\Delta g - H^0 \Delta q)^T}{(\Delta g - H^0 \Delta q)^T \Delta q}, \]

(23a)

\[ H_{\text{P}}^{k+1} = H^0 + \frac{(\Delta g - H^0 \Delta q)\Delta q^T + \Delta q(\Delta g - H^0 \Delta q)^T}{\Delta q^T \Delta q} \]

\[ - \frac{\Delta q^T (\Delta g - H^0 \Delta q) \Delta q \Delta q^T}{\Delta q^T \Delta q}, \]

(23b)

\[ \phi_k = 1 - \frac{(\Delta q^T (\Delta g - H^0 \Delta q))^2}{\Delta q^T \Delta q (\Delta g - H^0 \Delta q)^T \Delta q}. \]

(23c)

with \( \Delta q = q^{k+1} - q^k \) and \( \Delta g = G^{k+1} - G^k \). The Hessians are updated for a desired number of steps, before accurate Hessians are calculated.

III. NUMERICAL TESTS

To illustrate the efficiency of the above Hessian-based integration methods, the reactions \( \text{H}_2\text{CO} \rightarrow \text{H}_2 + \text{CO} \) \(^{42,55} \), \( +\text{C}_2\text{H}_5 \) \(^{56} \), and \( +\text{F} - +\text{CH}_3\text{OOH} \) \(^{47} \) were taken as test cases. \( \text{H}_2\text{CO} \rightarrow \text{H}_2 + \text{CO} \) dissociation has been used as a model in previous tests of integration schemes. \(^{52,55} \) The latter two reaction schemes were also investigated, since they have longer time dynamics and provide additional tests on the numerical integration. As part of these tests, the Hessian-based integrators are compared with both the Adams-Moulton \(^{1} \) and velocity-Verlet \(^{43} \) integration algorithms. All the calculations were carried out using a development version of \( \text{VENUS}^{58} \) interfaced with \( \text{NWChem-4.7} \) \(^{59} \).

A. \( \text{H}_2\text{CO} \rightarrow \text{H}_2 + \text{CO} \)

The simulation of \( \text{H}_2\text{CO} \rightarrow \text{H}_2 + \text{CO} \) is initiated at the dissociation transition state. Two trajectories were calculated, one starting with a reaction coordinate translational energy of 5.145 kcal/mol, zero-point energy in the other vibrational modes, and no rotational energy. The other has the same initial conditions, except there is a 298 K rotational energy of \( \text{RT/2} \) about each axis. The trajectories were then integrated on the \( \text{HF/6-31G(d,p)} \) \( \text{ab initio} \) potential, with a SCF convergence of \( 10^{-4} \) for the energy and gradient, and \( 10^{-6} \) for the Hessian evaluation. A trajectory was stopped when the distance between the fragments is greater than 6.0 Å. Each trajectory takes about 24 fs of atomic motion to satisfy these stopping criteria.

For the Hessian-based integration, the trust radius was updated as discussed above, with \( R_e = 0.2 \) Å, \( r_0 = 0.002 \) Å, \( \alpha_1 = 0.5, \alpha_2 = 1, \) and \( \beta = 1.44 \). The Hessians were updated every five, six, seven, or eight steps using Eq. (22). A range of integration step sizes was tested for the velocity-Verlet and Adams-Moulton algorithms.

The \( \text{H} - \text{H} \) bond distance as a function of time is compared in Fig. 3(a) for the trajectory without rotational energy calculated with the Hessian-based algorithm using seven Hessian updates, the velocity-Verlet algorithm using a step size of 0.3 fs, and the Adams-Moulton algorithm using a step size of 0.3 fs. (The trajectory with \( \text{RT/2} \) for energy for each rotational axis gives results very similar to those in Fig. 3.) The trajectory obtained from the Hessian-based integrator is nearly identical to the Adams-Moulton trajectory and matches well with the one from the velocity-Verlet integrator. The total energy of the system is plotted against the integration time in Fig. 3(b) for the three integrators. The total energy is conserved well by the Hessian-based and Adams-Moulton integrators, better than for the velocity-Verlet algorithm. Because of the symplectic nature of the velocity-Verlet integrator, periodicity is present in this integrator’s total energy. The average energy determined over one period of this regular motion is well preserved, even for the largest integration step size. The error in this average energy is quite small.

The Hessian-based integrator was also tested for a different trust radius updating parameter, \( R_e \), and different numbers of Hessian updates. The salient features of the results are listed in Table I for the trajectory with no rotational energy, calculated with the Hessian-based integrator. The results are nearly the same for the trajectory calculated with \( \text{RT/2} \) (\( T = 298 \) K) of energy in each rotational degree of free-
dom. It is seen that the Hessian-based integrator shows good energy conservation both in terms of accumulated error and in terms of significant digits in energy. The average error during the integration is less than 0.1%. As expected, the CPU time decreases as the number of Hessian updates, between the *ab initio* calculations of the Hessian, is increased.

An interesting result from the tests of the Hessian-based integrator is that the average error decreases as the number of Hessian updates, between *ab initio* calculations of the Hessian, is increased. This is a counterintuitive result since an *ab initio* Hessian is expected to be more accurate than an updated Hessian. The origin of this effect is related to the trust radii for the two simulations with different numbers of Hessian updates. The inaccuracy associated with using more Hessian updates results in an overall reduced length in the trust radius. This increases the accuracy of the trajectory which may have a somewhat larger effect than the reduced accuracy associated with the increased number of updates. It can be seen from Table I that the number of *ab initio* gradient calls remains the same with a change in the number of Hessian updates, while the CPU time decreases. This is due to a decrease in the number of *ab initio* Hessian calls with an increase in Hessian updates.

Properties of the Hessian-based integrator with general Cartesian coordinates are listed in Table II and they may be compared with the properties of the rotated coordinate inte-
time ratio for $F^- + CH_3OOH$ using B3LYP/6-31G(d) is estimated to be $\sim 5$. Thus, with one gradient evaluation identified as a 1 CPU time factor, there is a total CPU time factor of 12 for 7 fs of motion for the $F^- + CH_3OOH$ system. For $O_3 + C_3H_6$, the Hessian to gradient CPU time ratio is 14. For a time step size of 7 fs between two Hessian calls, this gives a CPU time factor of 21.

A typical time step for the velocity-Verlet algorithm is 0.3 fs to maintain the same degree of accuracy as obtained with 1 fs for the Hessian-based integrator. Since the velocity-Verlet algorithm has only one gradient call per step, the CPU time factor for 7 fs of motion is then 23. Comparing this time factor with those given above for the Hessian-based integrator, shows that at the B3LYP/6-31G(d) level of theory the velocity-Verlet algorithm is approximately two times slower for $F^- + CH_3OOH$, but approximately the same speed for $O_3 + C_3H_6$. It should be noted that these comparisons are only for two systems, one level of theory, i.e., B3LYP/6-31G(d) and the NWChem-4.7 software package.

The ratio of the Hessian to gradient CPU times is expected to depend on the electronic structure algorithm used, the system under investigation, and the level of theory. Evaluating the

<table>
<thead>
<tr>
<th>Integrator</th>
<th>Step size (fs)</th>
<th>$R_c$ (Å)</th>
<th>Number of Hessian updates$^b$</th>
<th>CPU time (s)</th>
<th>Significant digits</th>
<th>Average error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>H-B/rotated</td>
<td>0.2</td>
<td>5</td>
<td>31</td>
<td>173</td>
<td>3</td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>6</td>
<td>29</td>
<td>158</td>
<td>3</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>7</td>
<td>29</td>
<td>147</td>
<td>3</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>8</td>
<td>28</td>
<td>144</td>
<td>3</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>5</td>
<td>27</td>
<td>162</td>
<td>3</td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>6</td>
<td>27</td>
<td>152</td>
<td>3</td>
<td>0.13</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>7</td>
<td>25</td>
<td>140</td>
<td>3</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>8</td>
<td>26</td>
<td>134</td>
<td>3</td>
<td>0.03</td>
</tr>
<tr>
<td>Velocity-Verlet</td>
<td>0.1</td>
<td></td>
<td>240</td>
<td>730</td>
<td>2</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td></td>
<td>120</td>
<td>374</td>
<td>2</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td></td>
<td>80</td>
<td>265</td>
<td>1</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td></td>
<td>60</td>
<td>204</td>
<td>1</td>
<td>1.2</td>
</tr>
<tr>
<td>Adams-Moulton</td>
<td>0.1</td>
<td></td>
<td>480</td>
<td>1086</td>
<td>4</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td></td>
<td>240</td>
<td>618</td>
<td>3</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td></td>
<td>160</td>
<td>458</td>
<td>2</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td></td>
<td>120</td>
<td>379</td>
<td>1</td>
<td>1.15</td>
</tr>
</tbody>
</table>

$^a$Number of Hessian updates between $ab$ initio calculations of the Hessian.
$^b$Total number of $ab$ initio calls for the trajectory. The gradient is calculated at each call. The Hessian is only calculated after the specified number of updates.
$^c$The CPU time is for a single 3.2 GHz Xeon processor with 4 Gbytes of random access memory (RAM).
$^d$Total energy is 17.918 kcal/mol.
The roles of these constituents, on the efficiencies of different integration algorithms, will be the focus of future studies.

IV. SUMMARY AND CONCLUSIONS

In the work presented here a previously proposed predictor-corrector Hessian-based integrator \(^{42}\) is modified to make it more accurate and computationally efficient. The integration uses a local quadratic expansion of the potential which is assumed to be accurate within a “trust radius.” The modified integration algorithm evaluates the trajectory for both the predictor and corrector steps in Cartesian coordinates. The previous algorithm uses instantaneous normal mode coordinates for the predictor step and rotated Cartesian coordinates for the corrector step. Since angular momentum is not explicitly represented in the instantaneous normal mode Hamiltonian and the local quadratic expansion of the potential is not rotationally invariant, the previous algorithm does not rigorously conserve angular momentum. In addition to this improvement, a new trust radius updating algorithm that uses the difference between the trajectories of the predictor and corrector steps is presented.

The complete Hessian-based integration algorithm is summarized as a flowchart in Fig. 4. A second-order potential is used for integration in the prediction step up to the end of the trust radius. At the end of the trust radius an \textit{ab initio} potential, gradient, and Hessian are calculated, which are then used to fit the trust region to a fifth-order polynomial potential. Bofill’s Hessian updating scheme is used to reduce the computational cost of the Hessian evaluation. Updating the trust radius provides additional computational enhancement. The Hessian-based integration algorithm was tested using the \(\text{H}_2\text{CO} \rightarrow \text{H}_2 + \text{CO}, \text{O}_3 + \text{C}_3\text{H}_6,\) and \(\text{F}^- + \text{CH}_3\text{OOH}\) reactions and compared with the velocity-Verlet integration algorithm. Tests revealed that this fifth-order Hessian-based integrator conserves the energy, angular momentum, and the trajectory motion quite well. With this integrator, the average error in energy during the integration is less than 0.1% and for comparable energy conservation the integrator is at least two times faster than the velocity-Verlet algorithm for \(\text{H}_2\text{CO} \rightarrow \text{H}_2 + \text{CO}\) and \(\text{F}^- + \text{CH}_3\text{OOH}\.\) The Hessian-based integrator has been incorporated into the \textsc{venus/nwchem} package. This is freely available software and will be made available to the research community. It should be noted that an analytic Hessian is required for this interpolation/integration algorithm to be computationally practical.

The overall logic of the current interpolation scheme

<table>
<thead>
<tr>
<th>System</th>
<th>CPU time (s)(^a)</th>
<th>CPU time ratio Hessian/gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{F}^- + \text{CH}_3\text{OOH})</td>
<td>133.0</td>
<td>5.1</td>
</tr>
<tr>
<td>(\text{O}_3 + \text{C}_3\text{H}_6)</td>
<td>164.2</td>
<td>14.3</td>
</tr>
</tbody>
</table>

\(^a\)The calculations were performed on a single 3.2 GHz Xeon processor with 4 Gbytes of RAM.

TABLE III. Gradient and Hessian CPU times, and their ratio, required for the \(\text{F}^- + \text{CH}_3\text{OOH}\) and \(\text{O}_3 + \text{C}_3\text{H}_6\) systems. The calculations were performed at the B3LYP/6-31 G(d) level of theory using \textsc{nwchem} \(^{59}\) (Ref. 59). The chemical dynamics simulations for \(\text{F}^- + \text{CH}_3\text{OOH}\) and \(\text{O}_3 + \text{C}_3\text{H}_6\) are presented in Ref. 57 and 56, respectively.

The calculations were performed on a single 3.2 GHz Xeon processor with 4 Gbytes of RAM.

FIG. 4. Flowchart representation of the complete Hessian-based integrator.
has some similarities to others which have been proposed. However, an essential component of the current interpolation is that it gives the correct trajectory for the specified initial condition. In future work it would be of interest to compare the computational efficiency of our interpolation scheme with the other proposed schemes, given the requirement they give the correct trajectory. It is expected that very fine interpolation will be needed to obtain the correct trajectory. This requirement may be critical if some of the trajectory initial conditions consist of regular, quasiperiodic motion, since inaccuracies in the trajectory integration could convert these regular trajectories to irregular ones with chaotic motion. The importance of obtaining the correct trajectory is emphasized by a recent direct dynamics simulation by Bach et al., in which ~20% of the trajectories for a microcanonical ensemble of highly excited C2H5 radicals consisted of regular-type trajectories, an intrinsic non-Rice-Ramsperger-Kassel-Marcus result. On the other hand, if the ensemble of trajectories has only irregular/chaotic motion, this strict requirement of obtaining the correct trajectory is unnecessary, since the noisy chaotic trajectories map onto each other.

In future work, several extensions can be made to the work presented here. The current interpolation algorithm only uses \textit{ab initio} data calculated during the immediate integration step. It should be possible to use additional \textit{ab initio} data points, which lie close to those for the immediate integration step and which were obtained from previous integration steps of the current trajectory and/or previous trajectories in the ensemble, to construct a more extensive interpolated region of the PES with a larger trust radius. This would further enhance the calculation of the trajectory by reducing the number of integration steps. In addition, the possibility of using \textit{ab initio} points, from previous integration steps of the current trajectory and those from previous trajectories in the ensemble, to interpolate a locally accurate PES for the immediate integration step, without additional \textit{ab initio} calculations, should also be investigated. The interpolation schemes proposed in earlier work should be considered, but as discussed above it is important for the interpolation scheme to give the correct trajectory for the specified initial condition.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant Nos. CHE-0124677 and CHE-0615321, the Robert A. Welch Foundation under Grant No. D-0005, and the MSCF/PNNL Computational Grand Challenge Project GC3556. The authors thank Srirangam Addepalli and the High-Performance Computing Center at Texas Tech University for their support.

51 In Ref. 42 different sets of interpolating polynomials are used which are equivalent to the one used in the present study. One of the polynomials in Ref. 42, \( y_5(u) = s(4u^3 + 7u^4 - u) \), should be corrected as 
\( y_5(u) = s(-4u^3 + 7u^4 - 3u) \).