A climbing string method for saddle point search

Weiqing Ren[1,a)] and Eric Vanden-Eijnden[2,b)]
[1]*Department of Mathematics, National University of Singapore, Singapore and Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore*
[2]*Courant Institute of Mathematical Sciences, New York University, New York, New York 10012, USA*

(Received 10 January 2013; accepted 12 March 2013; published online 1 April 2013)

The string method originally proposed for the computation of minimum energy paths (MEPs) is modified to find saddle points around a given minimum on a potential energy landscape using the location of this minimum as only input. In the modified method the string is evolved by gradient flow in path space, with one of its end points fixed at the minimum and the other end point (the climbing image) evolving towards a saddle point according to a modified potential force in which the component of the potential force in the tangent direction of the string is reversed. The use of a string allows us to monitor the evolution of the climbing image and prevent its escape from the basin of attraction of the minimum. This guarantees that the string always converges towards a MEP connecting the minimum to a saddle point lying on the boundary of the basin of attraction of this minimum. The convergence of the climbing image to the saddle point can also be accelerated by an inexact Newton method in the late stage of the computation. The performance of the numerical method is illustrated using the example of a 7-atom cluster on a substrate. Comparison is made with the dimer method. © *2013 American Institute of Physics*. [http://dx.doi.org/10.1063/1.4798344]

I. INTRODUCTION

Locating saddle points on a multidimensional potential energy surface is a challenging computational problem. It is of great interest in the context of reactive processes involving barrier crossing events. Indeed, the transition state theory asserts that these saddle points are the dynamical bottlenecks for the transitions, thereby explaining their mechanisms and allowing one to compute their rates. A number of methods have been proposed to compute the transition pathways and saddle points when both the initial state and the final state of a transition are known.[1–8] The situation becomes more difficult when the final state of the transition is unknown, and the aim is to find saddle points on the boundary of the basin of attraction around a given minimum using the minimum as the only input. To this end, a number of computational techniques have been designed, which include eigenvector following approaches, the activation relaxation technique (ART), the dimer method, the one-side growing string method (OGS), the shrinking dimer method, the gentlest ascent dynamics (GAD), etc.[9–25] Accelerated molecular dynamics techniques, which use the initial state as the only input, were also proposed for the study of transition events.[26–29]

The aim of the present paper is to propose a simple modification of the string method,[7,8] termed the climbing string method (CSM), to perform these computations. As in the original string method, CSM evolves a continuous curve with a fixed parametrization (the string) on the potential energy surface. One end point of the string is fixed at the minimum, while the other end point (the climbing image) is evolved uphill according to a modified force in which the component of the potential force in the tangent direction of the string is reversed. In the meanwhile, the string connecting the two end points follows the steepest descent dynamics and relaxes to the minimum energy path (MEP) connecting these end points. To prevent the climbing image of the string from escaping from the basin of attraction of the minimum and converging to irrelevant saddle point (i.e., saddle points not directly connected to the minimum), we examine the energy along the string at each time step or once in a few time steps. If the energy becomes non-monotonic, which is an indication that the climbing image of the string might have escaped from the basin, we truncate the string at the first maximum of the energy along it. The state at the maximum then becomes the new climbing image. The process repeats until convergence, when the string becomes a MEP connecting the minimum to a saddle point on the boundary of its basin of attraction.

Besides its simplicity, the main advantage of CSM is that the saddle points it locates are guaranteed to lie on the boundary of the basin of the given minimum. This is in contrast to the methods listed above which evolve a single image on the energy surface. The image may escape from the basin of the starting minimum and converge to saddle points that are not located on the boundary of this basin. In these methods, an additional step is then needed to check which minima the saddle point is connected to.

CSM can also be combined with other methods to evolve the climbing image. For example, to accelerate the convergence, we can switch to an inexact Newton method after the climbing image has been evolved to the vicinity of a saddle point. In the inexact Newton method, this climbing image is used as initial guess, and the step vector is computed by approximately solving a symmetric indefinite linear system. The inexact Newton method typically requires a few iterations to achieve convergence to a saddle point with high accuracy.

a)Electronic mail: matrw@nus.edu.sg
b)Electronic mail: eve2@cims.nyu.edu

The remainder of this paper is organized as follows. In Sec. II, we review the string method for the computation of MEPs. In Sec. III, the string method is extended to saddle point search and we discuss the algorithmic details of CSM. The inexact Newton method is presented in Sec. IV. In Sec. V, we illustrate the performance of CSM using a system consisting of a seven-atom cluster on top of a substrate, and compare it to the dimer method. Concluding remarks are made in Sec. VI. Details of some algorithms used in the paper are given in Appendices A and B.

II. THE STRING METHOD

The (zero-temperature) string method was designed for the computation of MEPs.[7,8] Denote by $V(x)$ the potential energy of the system, and let a and b be the location of two minima of this potential. The MEP between a and b is a curve connecting these minima which is parallel to the potential force.

The string method finds the MEP using the steepest descent dynamics in the space of paths.[7] One starts with an initial string $\varphi(\alpha, 0) = \psi(\alpha)$, which connects a and b and is parameterized by $\alpha \in [0, 1]$, e.g., the normalized arc-length. The initial string can be constructed in various ways, e.g., the linear interpolation between a and b. The string is then evolved according to

$$\dot{\varphi} = -\nabla V(\varphi) + (\nabla V(\varphi), \hat{\tau})\hat{\tau} + \lambda\hat{\tau}, \quad 0 < \alpha < 1 \tag{1}$$

with the two end points fixed at a and b, respectively,

$$\varphi(0, t) = a, \quad \varphi(1, t) = b. \tag{2}$$

In (1), $\dot{\varphi}$ is derivative of φ with respect to time t; $\hat{\tau} = \varphi'/|\varphi'|$ is the unit tangent vector to the string, where φ' is derivative of φ with respect to α; $(\cdot,\ \cdot)$ denotes the inner product. The last term in (1) does not affect the evolution of the curve $\gamma(t) = \{\varphi(\alpha, t)\colon \alpha \in [0, 1]\}$, but it allows one to control its parametrization and the distribution of the images along the string after discretization: $\lambda(\alpha, t)$ is the Lagrange multiplier used to this end. In practice, the action of the term $\lambda\hat{\tau}$ can be accounted for easily using interpolation/reparametrization technique. The steady-state solution of (1) and (2) is a MEP connecting the two states a and b along which

$$(\nabla V)^{\perp}(\varphi) := \nabla V(\varphi) - (\nabla V(\varphi), \hat{\tau})\hat{\tau} = 0. \tag{3}$$

The simplified string method[8] is based on the observation that the second term at the right-hand side of (1), being parallel to $\hat{\tau}$, can be absorbed in the Lagrange multiplier term, i.e., (1) can also be written as

$$\dot{\varphi}(\alpha, t) = -\nabla V(\varphi) + \bar{\lambda}\hat{\tau}, \quad 0 < \alpha < 1, \tag{4}$$

where $\bar{\lambda}(\alpha, t)$ is a new Lagrange multiplier.

In practice, the string is discretized into a sequence of images, $\{\varphi_0^k, \varphi_1^k, \cdots, \varphi_N^k\}$, where $\varphi_i^k = \varphi(i\Delta\alpha, k\Delta t)$ is the ith image along the string at time $t = k\Delta t$, with $\Delta\alpha = 1/N$ and Δt being the time step. These images are evolved according to the following two-step procedure:

1. Evolve each image by the projected potential force $-(\nabla V)^{\perp}(\varphi_i^k)$ (in the original method) or the bare potential force $-\nabla V(\varphi_i^k)$ (in the simplified method);
2. Redistribute the images along the string using interpolation/reparametrization.

These two steps are repeated until the images reach a steady state.

In (2), the two end points of the string are fixed at a and b, respectively, during the evolution of the string. These conditions can be relaxed when we compute the MEP between two minima. In this case, we may choose an initial string with the two end points being in the basins of the two minima, respectively. Then the string is evolved according to the dynamics in (4), but with the boundary conditions in (2) replaced by

$$\dot{\varphi}(\alpha, t) = -\nabla V(\varphi(\alpha, t)), \quad \alpha = 0, 1. \tag{5}$$

This equation simply means that the two end-points are evolved towards the nearest minimum according to the steepest descent dynamics. Therefore, as long as the two end points of the initial string lie in the basins of attraction of two minima of interest, respectively, they will converge to the minima and the string will converge to a MEP connecting them.

Next, we show that, with another choice of boundary conditions for the string, the string method can be modified to find saddle points around a given minimum.

III. THE CLIMBING STRING METHOD FOR SADDLE POINT SEARCH

Let a be the location of a minimum of the potential $V(x)$. We are interested in finding the saddle points that are directly connected to a, i.e., saddle points that lie on the boundary of the basin of attraction of a. To this end we start with an initial string $\varphi(\alpha, 0) = \psi(\alpha)$, with one end point fixed at a, $\psi(0) = a$, and the other at an initial guess for the location of the saddle—this guess can be quite bad, e.g, by taking $\psi(1)$ close to a. We then evolve the string using (4) by keeping one end-point fixed at a,

$$\varphi(0, t) = a, \tag{6}$$

and using the following modified boundary condition at the other end:

$$\dot{\varphi}(1, t) = -\nabla V(\varphi) + \nu(\nabla V(\varphi), \hat{\tau})\hat{\tau}. \tag{7}$$

Here $\hat{\tau}$ is the unit tangent vector to the string at $\alpha = 1$ and ν is a parameter larger than 1. Equation (7) means that the potential force acting on the final point is reversed (then rescaled by $\tilde{\nu} = \nu - 1$) in the direction tangent to the string. This makes the final point climb uphill on the potential energy surface in this tangent direction while following the original steepest descent dynamics in the subspace perpendicular to it. The value of ν controls the ascent speed of the final point relative to the steepest descent dynamics in the perpendicular subspace. The larger the parameter ν, the faster the final point moves in the tangent direction. In the examples below we fixed $\nu = 2$.

Equation (4) together with the boundary conditions (6) and (7) forms a closed system which is solved with initial condition $\varphi(\alpha, 0) = \psi(\alpha)$ until a steady state is reached. At the steady state, the end point of the string moving according to (7) will converge to a saddle point, and the string will converge to the MEP connecting this saddle point and the minimum at a.

For complex systems, the final point of the string may escape from the basin of the minimum and converge to a saddle point which is not directly connected to the minimum. To prevent this from happening, we impose a constraint on the above dynamics and require the potential energy to be monotonically increasing along the string from the minimum at $\alpha = 0$ to the final point at $\alpha = 1$. This guarantees that the obtained saddle point lies on the boundary of the basin of the minimum a. In practice, the constraint can be easily imposed by truncating the string at the point where the potential energy attains the (first) maximum, followed by a reparametrization of the string using normalized arc-length.

Next, we describe in detail how the various steps described above are implemented in practice.

A. Step 1: Evolution of the string

As in the string method, the string is discretized into a sequence of images $\{\varphi_0, \varphi_1, \ldots, \varphi_N\}$, where $\varphi_i = \varphi(i\Delta\alpha, t)$ is the ith image along the string at time t and $\Delta\alpha = 1/N$. The intermediate images are then evolved over one time step Δt according to

$$\dot{\varphi}_i = -(\nabla V)^{\perp}(\varphi_i), \quad i = 1, \ldots, N-1, \tag{8}$$

where the projection of the force is defined in (3) and the tangent vectors $\hat{\tau}$ are computed using an upwinding finite difference scheme.[7] Alternatively, in the simplified string method, the bare potential force (without any projection) can also be used to evolve the intermediate images. Concerning the endpoints, the first image is being kept fixed at a, $\varphi_0 = a$ and the last image follows

$$\dot{\varphi}_N = -\nabla V(\varphi_N) + \nu(\nabla V(\varphi_N), \hat{\tau}_N)\hat{\tau}_N, \tag{9}$$

where $\nu > 1$ (e.g., $\nu = 2$) and $\hat{\tau}_N$ denotes the unit tangent vector at the final point of the string, which is computed using the one-sided finite difference,

$$\hat{\tau}_N = \frac{\varphi_N - \varphi_{N-1}}{|\varphi_N - \varphi_{N-1}|}, \tag{10}$$

where $|\cdot|$ denotes the Euclidean norm. Equations (8) and (9) can be integrated in time by any suitable ODE solver, e.g., the forward Euler method,

$$\varphi_i^* = \varphi_i^k - \Delta t\,(\nabla V)^{\perp}\big(\varphi_i^k\big), \quad i = 1, \ldots, N-1, \tag{11}$$

where $\varphi_i^k = \varphi(i\Delta\alpha, k\Delta t)$, and similarly for (9). Other ODE solvers, e.g., the Runge-Kutta methods, can be used as well.

The result of Step 1 is a new sequence of images, $\{\varphi_0^*,\ \varphi_1^*, \ldots, \varphi_N^*\}$. Note that Step 1, which is the most costly since it is the only one involving force calculations, can be trivially parallelized since all the images move independently of each other.

B. Step 2: Imposing the monotonic-energy constraint

The purpose of this step is to prevent the end point φ_N^k from converging to a saddle point which is not on the boundary of the basin of the minimum located at a. At each time step, the potential energy along the string, $\{V(\varphi_i^*),\ i = 0, 1, \ldots N\}$, is computed. If the sequence of the potential energy is monotonically increasing, then we keep the whole string and go to the reparametrization step; otherwise we denote by J the index at which the first maximum of the potential energy is attained, and define the new string by the truncated sequence

$$\{\varphi_0^*,\ \varphi_1^*,\ \ldots,\ \varphi_{J-1}^*\}. \tag{12}$$

C. Step 3: Reparametrization of the string

The purpose of this step is to impose the equal-arclength constraint (or other parametrization) so that the discrete images remain evenly distributed along the string. It can be done using interpolation techniques in a straightforward manner. It consists of two steps: first we interpolate across the discrete images obtained from Step 2, then we compute a set of new images on a uniform grid. Given the images $\{\varphi_i^*,\ i = 0, \ldots, M\}$ (where $M = J - 1$ if the string was truncated and $M = N$ otherwise), a continuous curve γ can be constructed by interpolating these images using, e.g., linear interpolation, splines, etc. This gives an analytic representation for γ in the form of $\gamma = \{\varphi(\alpha)\colon \alpha \in [0, 1]\}$ whose specific form depends on the parametrization we choose, such as the normalized arc-length. Using this continuous representation of the string, it is straightforward to obtain N new discrete images on a uniform grid,

$$\varphi_i^{k+1} = \varphi(i/N), \quad i = 0, 1, \ldots, N. \tag{13}$$

In the simplest case when we choose to enforce the equal arclength parametrization, the problem is simply that of interpolating given values $\{\varphi_i^*\}$ defined on a non-uniform mesh $\{\alpha_i^*,\ i = 0, \ldots, M\}$, onto a uniform mesh with N points. To do so, we first compute the arc-length at the given images,

$$s_0 = 0, \quad s_i = s_{i-1} + |\varphi_i^* - \varphi_{i-1}^*|, \tag{14}$$

for $i = 1, 2, \ldots, M$. The mesh $\{\alpha_i^*\}$ is then obtained by normalizing $\{s_i\}$:

$$\alpha_i^* = s_i/s_M, \quad i = 0, 1, \ldots, M. \tag{15}$$

The new images φ_i^{k+1} on the uniform grid $\{\alpha_i = i/N, i = 0, 1, \ldots, N\}$ are then computed by cubic polynomial or cubic spline interpolation across the points $\{(\alpha_i^*, \varphi_i^*), i = 0, 1, \ldots, M\}$.

At the end of Step 3, we have computed an updated sequence of equidistant images along the string, $\{\varphi_0^{k+1}, \varphi_1^{k+1}, \ldots, \varphi_N^{k+1}\}$, and we can go back to Step 1 for another iteration. This process is continued until convergence, which is monitored according to

$$\max\Big\{\max_{1<i<N}\big|(\nabla V)^{\perp}\big(\varphi_i^k\big)\big|, \big|\nabla V\big(\varphi_N^k\big)\big|\Big\} < \delta_s, \tag{16}$$

where the norms are the maximum norm and δ_s is a prescribed tolerance.

IV. ACCELERATION OF THE CONVERGENCE BY INEXACT NEWTON METHOD

After the final point of the string has been evolved to the vicinity of a saddle point, as measured by the error in (16), it may be useful to switch the computation to a second phase to accelerate the convergence. Specifically, we apply an inexact Newton method to improve the accuracy of the approximation to the saddle point. The final point of the string obtained in the first phase is used as the initial guess for the inexact Newton method. Note that in the second phase we evolve only a single image (the final point of the string), which is denoted by x below, but not the entire string.

The initial guess for the inexact Newton method is given by the final point of the string: $x_0 = \varphi_N$. At step k, the image is updated as follows:

$$x_{k+1} = x_k + p_k, \tag{17}$$

where the step vector p_k satisfies the condition:

$$|H_k p_k + \nabla V(x_k)| \le \eta |\nabla V(x_k)|, \tag{18}$$

where the norm is the Euclidean norm and $H_k = \nabla^2 V(x_k)$ is the Hessian of the potential energy at x_k, and η is a prescribed small parameter called the forcing parameter. The iteration is terminated when the maximum norm of the potential force falls below some tolerance δ_n: $|\nabla V(x_k)| < \delta_n$.

To find a step vector p_k satisfying the condition in (18), we approximately solve the following linear system:

$$H_k p_k = -\nabla V(x_k). \tag{19}$$

The Hessian matrix H_k in (19) is symmetric but in general indefinite since x_k is in the vicinity of a saddle point. We solve (19) using an iterative method based on Krylov subspaces and the LQ factorization.[30] This inner iteration is terminated when the relative residue falls below the prescribed forcing parameter η, i.e., when condition (18) is met. Details of the algorithm are given in Appendix A. The iterative method requires a matrix-vector multiplication of the form $H_k u$ for some vector u at each iteration. To avoid the computation of the Hessian matrix, we employ the standard practice and use a finite difference to approximate the product:[16,26]

$$H_k u = \frac{\nabla V(x_k + \varepsilon u) - \nabla V(x_k)}{\varepsilon}, \tag{20}$$

where ε is a small parameter ($\varepsilon = 10^{-3}$ in the numerical examples). The approximation only requires the evaluation of potential forces. More accurate approximation using central finite difference can be used as well.

It can be shown that, under mild condition on the smoothness of the potential energy near the saddle point x^*, the sequence of iterates generated by (17) converges to x^*, provided the initial guess x_0 is sufficiently close to x^* and η is sufficiently small.[31] The convergence rate depends on the choice of η. For example, if $\eta \to 0$ as $k \to \infty$, then the convergence is super-linear, and if $\eta = \mathcal{O}(|\nabla V(x_k)|)$ then the convergence is quadratic. In practice, it only takes a few steps for the image to converge to a saddle point with high accuracy.

V. NUMERICAL EXAMPLES

We illustrate the performance of the numerical method proposed above using the system of a seven-atom cluster on a substrate (see Fig. 1). The cluster consists of seven atoms. The substrate is modeled by six layers of atoms with each layer containing 56 atoms. Periodic boundary conditions are used in the x and y directions, respectively. The system has been extensively studied in the literature[21] and it has a large number of saddle points.

The interaction between the atoms is modeled by the pairwise Morse potential:

$$V(r) = D(e^{-2\alpha(r-r_0)} - 2e^{-\alpha(r-r_0)}), \tag{21}$$

where r is the distance between the atoms, $D = 0.7102$, $\alpha = 1.6047$, and $r_0 = 2.8970$. Figure 1 shows the configuration of the system at the minimum of the potential energy.

The numerical methods will be illustrated using three examples. In the first example in Sec. V A, we consider a three-dimensional problem in which only the lower-left edge atom in the cluster is free to move. We use this system to illustrate the performance of the climbing string method and its convergence region. The second example in Sec. V B is used to illustrate the local convergence of the inexact Newton method. In the third example in Sec. V C, we apply the complete algorithm (the climbing string and the inexact Newton method) to compute the saddle points starting from the minimum, and compare the performance of the algorithm with the dimer method.

A. Performance of the climbing string method

For illustrative purpose, we first consider a simple case in which only the lower-left atom in the cluster is free to move while all the other atoms are fixed in their minimum configuration. The dimension of this problem is 3.

The convergence history of the string projected onto the xy plane is shown in Fig. 2. The linear interpolation

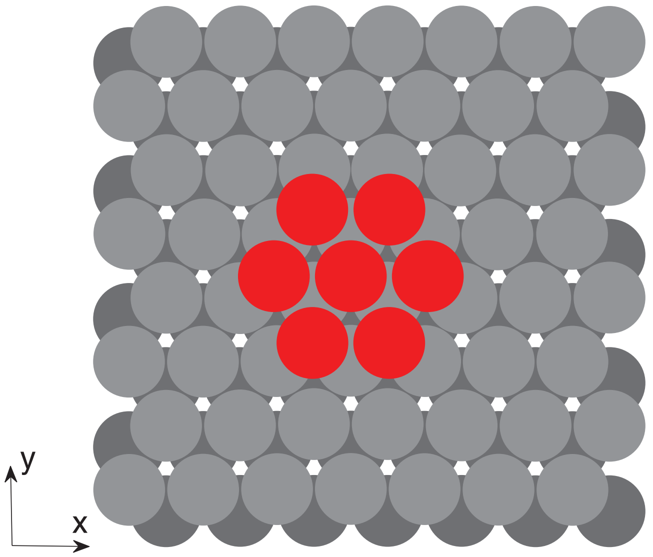

FIG. 1. The configuration of the seven-atom cluster on a substrate at the minimum of the potential energy.

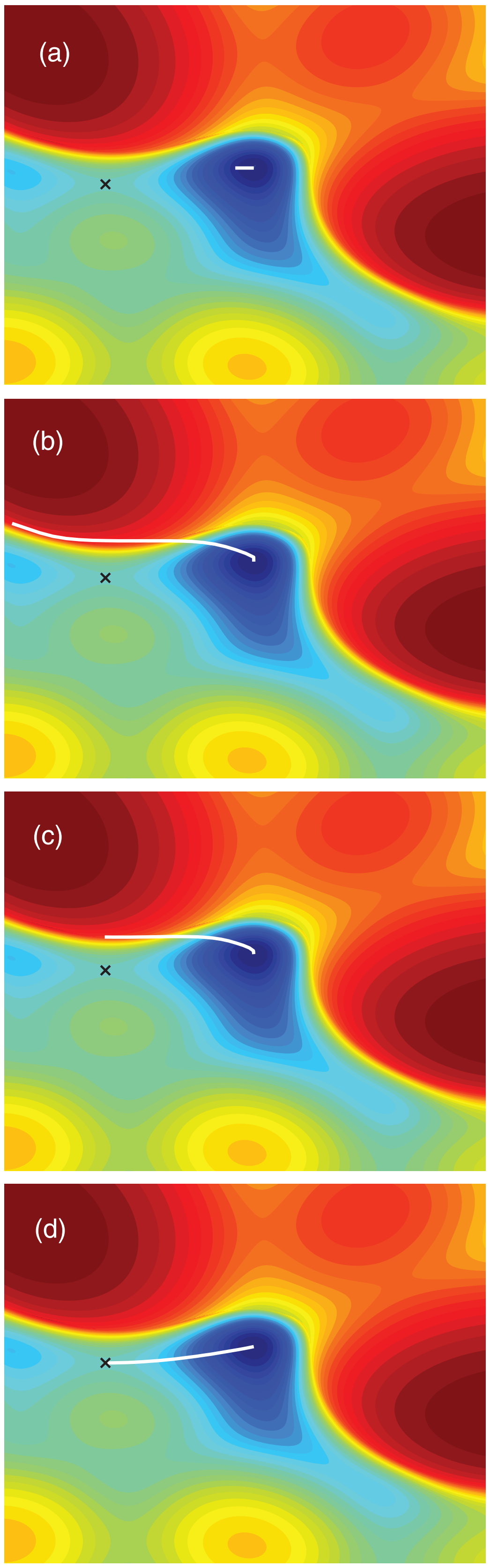

FIG. 2. Snapshots of the climbing string at different times. The string is projected onto the xy plane. The surface is the energy $\tilde{V}(x, y) = \min_z V(x, y, z)$. (a) The initial string; (b) the string at a time when the potential energy along the string becomes non-monotone; (c) the string after truncation at the point where the potential energy attains its first maxima; (d) the converged string and the saddle point located by the final point of the string.

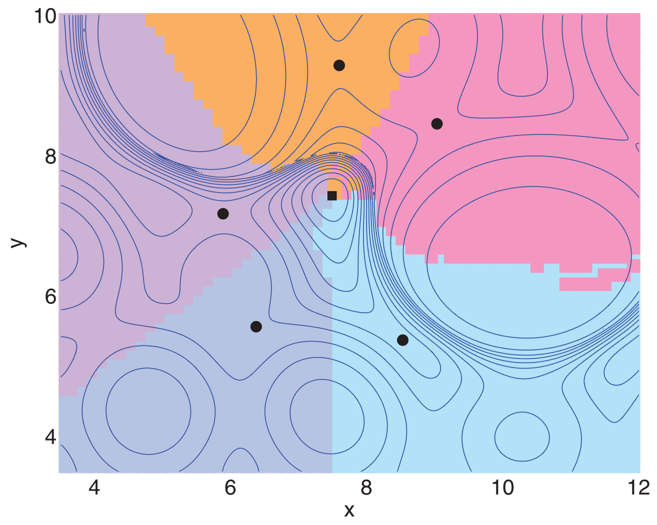

FIG. 3. The five saddle points (filled circles) and their corresponding convergence region in the 3D problem. The minimum is shown as the filled square near the center of figure. The lines are the level curves of the energy $\tilde{V}(x, y) = \min_z V(x, y, z)$.

between the minimum and an image obtained from a perturbation of the minimum is used as initial string which is shown in Fig. 2(a). Figure 2(b) shows the string at a later time when the potential energy along the string becomes non-monotonic, which indicates the end point of the string might have escaped from the basin of the minimum. The string is then truncated at the maxima of the potential energy along it. Figure 2(c) shows the string after the truncation. Finally, Fig. 2(d) shows the saddle point that the final point of the string converged to and the MEP between the minimum and the saddle point.

The system has 5 saddle points around the minimum. The convergence regions of these saddle points are shown in Fig. 3. The initial string is constructed by the linear interpolation between the minimum (the square near the center of the figure) and a perturbed state from the minimum. The string is evolved according to the climbing string algorithm and it converges to one of the five saddle points (the filled circles in Fig. 3). The five regions indicated by the solid colors are the convergence regions of the five saddle points, respectively. When the (x, y) coordinates of the final point of the initial string lies in one of the five regions (its z coordinate is chosen to minimize the potential energy), it converges to the corresponding saddle point within the region. Note that the convergence region depends on the choice of the initial string. In the present example, we used linear interpolation to construct the initial string. We also note that the convergence regions are larger than the basin of attraction of the minimum. Indeed, the climbing string method does not require the initial data to be within the basin of attraction for convergence to a saddle point on the boundary of the basin.

B. Performance of the inexact Newton method

The performance of the inexact Newton method is illustrated using a high dimensional example. In this example, the atoms in the 7-atom cluster and those in the top three layers of the substrate are free to move, while the other atoms are frozen in space. The system has dimension 525 in this case.

Since our purpose here is to illustrate the local convergence of the inexact Newton method, the initial guess is prepared by a random perturbation of a pre-computed saddle point. Each atom in the system is randomly displaced from the saddle point configuration by an amount drawn from the uniform distribution on [0, 0.1]. We carried out 25 runs from the perturbation of 25 different saddle points.

Besides the parameter ε used in the finite difference approximation of the Hessian-vector product (20), which takes the value 10^{-3} in this work, the other only parameter we need to prescribe is the forcing parameter η. The forcing parameter controls the accuracy of the solution to the linear system (19). With $\eta = 0.01$, the inexact Newton method takes an average of 189 force evaluations in 5 iterations to reduce the maximum norm of the force $|\nabla V|$ to below $\delta_n = 10^{-6}$. A smaller value of η requires fewer Newton iterations. On the other hand, each Newton iteration needs more force evaluations since it requires a more accurate solution of the linear system with a smaller η. As a result, the total number of force evaluations slightly increases when η is decreased. This is shown in Fig. 4, where the error (measured by the magnitude of the potential force) is plotted against the number of force evaluations for the convergence to the 25 different saddle points. The numerical results are obtained using different choices of η: $\eta = 0.1$ (circles), $\eta = 0.01$ (stars), and $\eta = 0.001$ (squares). Overall, the computational cost in terms of the number of force evaluations is not very sensitive to the value of η. In this example, two orders of difference in η only changes the average number of force evaluations by less than a factor of 2 - from $N_{Newton} = 158$ for $\eta = 0.1$ to $N_{Newton} = 276$ for $\eta = 0.001$.

C. Performance of the complete algorithm

In this example, we start from the minimum configuration of the 7-atom cluster system and compute saddle points around the minimum in the 525D space using the accelerated climbing string method. The computation proceeds as follows:

1. *Preparation of the initial data*. The configuration of the minimum is perturbed by randomly displacing the atoms in the cluster by Δx. This gives the final point of the initial string. The initial string is then obtained by linearly interpolating the minimum and the final point. The string is discretized using N points. N ranges from 5 to 20.
2. *Evolution of the string*. The climbing string method is applied to evolve the string. The time step $\Delta t = 0.03$ is used in this example. The string is reparametrized every 10 time steps. The evolution of the string is terminated when the condition in (16) is met, i.e., when the force on the string falls below $\delta_s = 0.01$.
3. *Acceleration by the inexact Newton method*. The final point of the string from the previous step is used as the initial data for the inexact Newton method. The forcing parameter $\eta = 0.01$. The computation is terminated when the maximum norm of the force falls below $\delta_n = 10^{-6}$.

The performance of the above algorithm is shown in Table I. The data in each row are based on 100 runs from different initial data. With $N = 20$ points along the string, each run successfully converged to a saddle point directly connected to the minimum, as indicated by the success ratio $\varrho = 1$ in the table. The success ratio decreases for smaller N, due to the discretization error and a poor representation of the string with the few number of points. However, even with as few as $N = 5$ points, we are still able to achieve a success ratio 0.96 (i.e., 96 of the 100 runs converged to a saddle point directly connected to the minima) for $\Delta x = 0.1$ and 0.71 for $\Delta x = 0.5$.

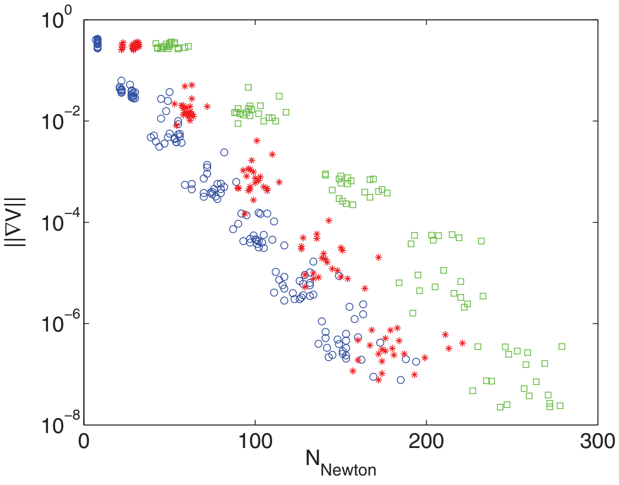

FIG. 4. Convergence history of the inexact Newton method to different saddle points and for different choice of the forcing parameter η: $\eta = 0.1$ (circles), $\eta = 0.01$ (stars), and $\eta = 0.001$ (squares). N_{Newton} is the number of force evaluations. Each cluster corresponds to one inexact Newton iteration.

TABLE I. Performance of the accelerated climbing string method for the example of 7-atom cluster in the 525d space. The performance of the dimer method is shown for comparison. The data are based on 100 runs from different initial data, which are prepared by randomly displacing the minimum by Δx. N is the number of images along the string, ϱ is the ratio of the number of runs that converged to a saddle point directly connected to the minimum, and n_s is the number of different saddle points obtained in these runs. N_{string} is the average number of steps (i.e., the number of force evaluations per image) in the climbing string method, N_{Newton} is the average number of force evaluations in the inexact Newton method, $N_{total} = N * N_{string} + N_{Newton}$ is the total number of force evaluations. N_{dimer} and N_{relax} are the average number of force evaluations in the dimer method and in the relaxation step, respectively.

Method	N	ϱ	n_s	N_{string}	N_{Newton}	N_{total}
String	5	0.96	20	475	241	2616
$\Delta x = 0.1$	8	0.98	25	468	244	3988
	16	0.97	9	493	259	8147
	20	1.00	8	507	266	10406
String	5	0.71	21	773	229	4049
$\Delta x = 0.5$	8	0.90	38	741	221	6149
	16	0.93	31	648	227	10595
	20	1.00	27	619	213	12592
	Δx	ϱ	n_s	N_{dimer}	N_{relax}	N_{total}
Dimer	0.1	0.40	24	5522	1490	7012
	0.5	0.43	28	4761	1772	6533
Dimer (CG)	0.1	0.31	19	1551	1660	3211
	0.5	0.39	30	1190	1606	2796

The last three columns show the average number of force evaluations per saddle point: N_{string} is the number of force evaluations per image (i.e., the number of time steps) in the first phase of the computation, N_{Newton} is the number of force evaluations in the acceleration phase by Newton method, and the last column is the total number of force evaluations for the whole string, i.e., $N_{total} = N * N_{string} + N_{Newton}$. On average, it requires several hundred steps to evolve the final point of the string from the neighborhood of the minimum to the vicinity of a saddle point in the first phase, and another several hundred force evaluations to improve the accuracy of the approximation in the second phase. These numbers are insensitive to the number of images that are used to discretize the string. When the number of images N increases, the computation becomes more costly. In fact, the total number of force evaluations for the whole string increases linearly with the number of images. However, we note that the string method can be easily parallelized and the force of the different images along the string can be computed simultaneously on different processors. Therefore, when implemented on a parallel computer with N processors, the efficiency of the algorithm is determined by the number of force evaluations per image (i.e., $N_{string} + N_{Newton}$) instead of N_{total}.

The climbing string method tends to locate low-barrier saddle points. Using a relatively small displacement $\Delta x = 0.1$ in the preparation of the initial data, 57 of the 100 strings ($N = 20$) converged to saddle points with the lowest barrier $\Delta V = 0.6011$, 37 strings converged to saddle points with the second lowest barrier $\Delta V = 0.6195$, and the rest converged to saddle points of barrier about $\Delta V \approx 1.6$. We obtained 8 different saddle points from the 100 runs.

By increasing the magnitude of the perturbation to $\Delta x = 0.5$, the number of different saddle points obtained using the climbing string method with $N = 20$ increased to 27. The histogram of the energy barriers obtained from the 100 runs is shown in Fig. 5 (upper panel). Similar to the previous result, the climbing string method finds a saddle point of low energy barrier with high probability. In Ref. 22, it was reported that ART and OGS methods also found lower lying saddle point preferably in a given direction in the example of the diffusion of a water molecule on NaCl(001) surface.

The performance of the climbing string method is compared with the dimer method in Table I. The dimer method is applied to the same set of initial data as in the climbing string method. The initial orientation of the dimer is chosen randomly. The dimer method has 6 parameters. In the computation, we used the values suggested in Ref. 21 (see Appendix B).

The dimer method has no guarantee that the computed saddle point lies on the boundary of the basin of the minimum we started with. An additional step is needed at the end of each run to check to which minima the obtained saddle point is connected. Therefore, the complete algorithm consists of two steps. In the first step, the system is evolved towards a saddle point following the eigenvector corresponding to the lowest eigenvalue of the Hessian; the algorithm is given in Appendix B. This step is terminated when the potential force falls below $\delta_n = 10^{-6}$. In the second step, the system is slightly perturbed from the computed saddle point in the unstable direction (forward and backward), then it is relaxed to the nearest minimum following the steepest descent dynamics. The steepest descent dynamics is solved using the forward Euler method with time step $\Delta t = 0.03$. The relaxation is terminated when the distance of the system to the minimum that we began with falls below 0.1 or the force falls below 10^{-3}. In the former case the saddle point is directly connected to the minimum and the search is successful, while in the later case it is not. For the 100 runs, the success ratio is $\varrho = 0.4$ when the initial perturbation $\Delta x = 0.1$ and $\varrho = 0.43$ when $\Delta x = 0.5$. About half of the searches converged to saddle points that are not directly connected to the minimum or failed to converge.

The columns N_{dimer} and N_{relax} in Table I show the number of force evaluations in the two steps, respectively. These numbers were averaged over the runs that successfully converged to a saddle point. The last column is the total number of force evaluations: $N_{total} = N_{dimer} + N_{relax}$.

The histogram of the energy barriers obtained from the successful searches (i.e., the runs that converged to a saddle point directly connected to the minimum) with $\Delta x = 0.5$ is shown in Fig. 5 (lower panel). Compared to the histogram obtained from the climbing string method, the distribution is

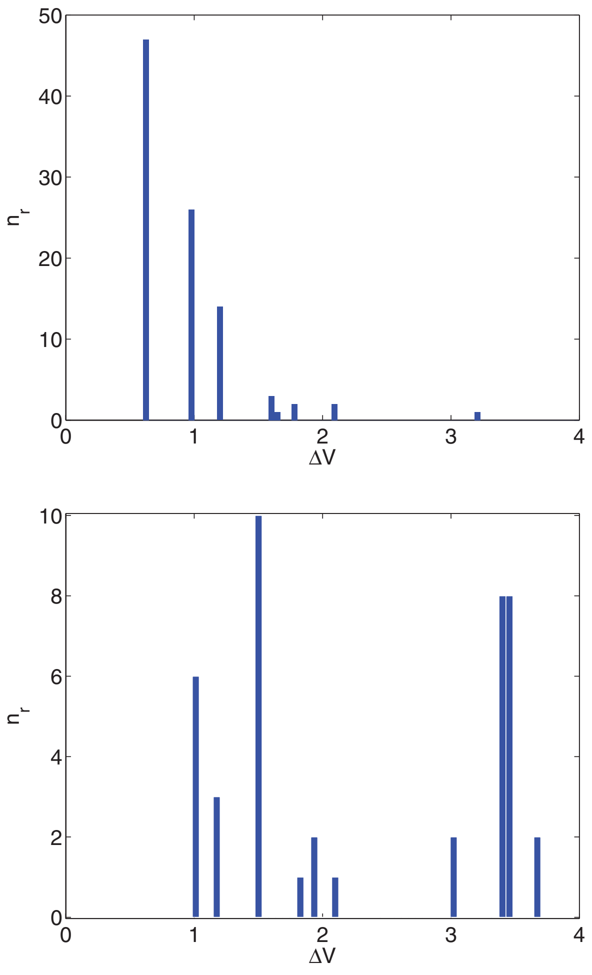

FIG. 5. Histogram of the energy barriers at the saddle points obtained from 100 runs using the climbing string method (upper panel, $N = 20$) and the dimer method (lower panel). The initial data were obtained by randomly displacing the minimum by $\Delta x = 0.5$. The climbing string method locates a saddle point of low energy with higher probability. Other runs with fewer images along the string exhibit similar behavior.

more random, and the result is more sensitive to the initial data. With the perturbation $\Delta x = 0.1$, 8 of the 100 searches found the lowest energy barrier $\Delta V = 0.6011$ (not shown), while with $\Delta x = 0.5$, none of the 100 searches found the lowest energy barrier as seen in the histogram.

As pointed out in Ref. 20, the efficiency of the dimer method can be improved by using conjugate gradient (CG) directions in the evolution of the dimer. Indeed, we found that using conjugate gradient directions can reduce the total number of force evaluations N_{total} by a half. The algorithm is given in Appendix B. The success ratio slightly reduces to $\varrho = 0.31$ for $\Delta x = 0.1$ and $\varrho = 0.39$ for $\Delta x = 0.5$. More than half of the 100 runs either converged to a saddle point not directly connected to the minimum or failed to converge.

VI. CONCLUSION

In this paper the string method originally proposed for the computation of minimum energy paths was extended to the computation of saddle points around a given minimum. The numerical method consists of evolving a string on the potential energy landscape with one end point fixed at the minimum, while the other end point (the climbing image) climbs uphill towards a saddle point until the potential force falls below a threshold. We also showed how the convergence of the final state of the string to the saddle point can be accelerated using an inexact Newton method in the final stage of the computation.

Compared to the existing methods for saddle points search, CSM uses a string instead of one image. This gives us control on the evolution of the climbing image based on the potential energy along the string. When the string is discretized using enough images, the saddle point located by the final state of the string is guaranteed to be directly connected to the minimum, i.e., it lies on the boundary of the basin of this minimum. In contrast, all the existing numerical methods based on one image have the difficulty that they may converge to irrelevant saddle points that are not directly connected to the given minimum.

There is no doubt that evolving a string is more costly than the methods using only one image. However, the method can be easily parallelized since the images along the string are only weakly coupled to each other. In particular, the computation of the forces and the energies of the different images along the string can be carried out on different processors. In addition, the evolution of the string with multiple images is only run for a short time and it is to provide good initial data for the more efficient Newton method. Once the final state of the string is evolved to a vicinity of a saddle point, the computation is then accelerated by the Newton method. In the second phase, only the final state of the string is evolved.

CSM is simple and easy to implement. One essential parameter that the user needs to prescribe is the tolerance δ_s for the termination of the evolution of the climbing string. The choice of δ_s is problem-dependent, and it relies on the characteristics of the potential energy near the saddle point. In general, the smaller the parameter δ_s, the more likely the inexact Newton method is to converge. Of course, a smaller δ_s requires more iterations and more force evaluations. In the inexact Newton method, the user also needs to specify the forcing parameter η which controls the accuracy of the approximate solution to the linear system in the inexact Newton method. However, our numerical examples show that the computational cost in terms of force evaluations is insensitive to the choice of this forcing parameter.

Our numerical examples also show that CSM has a large convergence region. In particular, it does not require the initial data to be within the basin of attraction of the minimum. CSM also tends to locate low-barrier saddle points. These saddle points are more relevant to noise-induced transitions out of the basin of the minimum. In contrast, the existing numerical methods, such as the dimer method, are more sensitive to the choice of the initial data.

Two interesting questions are how to avoid the convergence of the string to a saddle point that has already been found, and how to systematically enumerate as many saddle points as possible. This second goal could, for instance, be achieved by letting the climbing image make a random walk (or a biased random walk) on the potential energy surface while keeping the other components of the algorithm unchanged, in particular, truncating the string at the first maximum of the energy along it to prevent the climbing image from escaping the basin of the minimum of interest. Another question is how to choose the number of images and how to distribute these images along the string in order to optimize the efficiency of the numerical method. These issues will be studied in future work.

ACKNOWLEDGMENTS

We are grateful to Weinan E for helpful discussions. The work of Ren was in parts supported by Singapore A*STAR SERC "Complex Systems Programme" grant R-146-000-171-305 (Project No. 1224504056) and A*STAR SERC PSF grant R-146-000-173-305 (Project No. 1321202071). The work of Vanden-Eijnden was in parts supported by NSF (Grant No. DMS07-08140) and ONR (Grant No. N00014-11-1-0345).

APPENDIX A: ALGORITHM FOR SOLVING SYMMETRIC INDEFINITE LINEAR SYSTEMS

The linear system in (19) is solved using an iterative method. The algorithm is based on the Lanczos iteration and the *LQ* factorization of tridiagonal matrices.[30] To simplify the notation, let us write the linear system as $Hp = f$, where H is a symmetric matrix, and without loss of generality we assume $|f| = 1$. Starting with an initial guess $p^{(0)}$ ($p^{(0)} = 0$ was used in the numerical examples), the iterative method generates a sequence of iterates $\{p^{(1)}, p^{(2)}, \ldots, p^{(n)}, \ldots\}$. These approximate solutions are computed from the successive Krylov subspaces of H generated by f.

At the nth iteration, the matrix H is first reduced to a symmetric $n \times n$ tridiagonal matrix $T^{(n)}$:

$$V^{(n)^T} H V^{(n)} = T^{(n)}. \tag{A1}$$

This triangularization is done by the standard Lanczos iteration in which the vectors $\{v_1, v_2, \ldots, v_n\}$ forming the columns of $V^{(n)}$, and the scalars $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ and $\{\beta_2, \beta_3, \ldots, \beta_n\}$

forming, respectively, the diagonal and sub-diagonal of $T^{(n)}$ are obtained from

$$\begin{aligned} &v_0 = 0, \quad v_1 = f/\beta_1, \quad \beta_1 = |f|, \\ &w = Hv_j - \alpha_j v_j - \beta_j v_{j-1}, \quad \alpha_j = v_j^T H v_j, \\ &v_{j+1} = w/\beta_{j+1}, \quad \beta_{j+1} = |w|, \end{aligned} \tag{A2}$$

for $j = 1, 2, \ldots$. The approximate solution $p^{(n)}$ is found by first solving the equation

$$T^{(n)} y^{(n)} = \beta_1 e_1, \tag{A3}$$

where $e_1 = (1, 0, \ldots, 0)^T$, then setting

$$p^{(n)} = V^{(n)} y^{(n)}. \tag{A4}$$

The tridiagonal system (A3) is solved using the *LQ* factorization.

Following the above iterative procedure, we obtain a sequence of approximate solutions to (19). The iteration is terminated when the relative residue of $p^{(n)}$ falls below the threshold η as required in (18).

As a byproduct of the above procedure, we may compute the approximate extreme eigenvalues and the corresponding eigenvectors for the Hessian H, in particular the lowest eigenvalue and the corresponding eigenvector, using the tridiagonal matrix $T^{(n)}$. This strategy was used in the activation-relaxation technique[18,19] to compute the unstable direction near the saddle point. The system climbs uphill on the potential energy surface in the unstable direction while following the relaxation dynamics in the perpendicular directions. In the algorithm developed in this paper, the Lanczos iteration (A2) is used in different purpose—the aim here is to compute the step vector p in the inexact Newton method.

The algorithm for solving (19) is as follows.

Algorithm A (Lanczos iteration, symmetric LQ):

```
$p_0 = 0, v_0 = 0, v_1 = f, \bar{w}_0 = 0, \beta_1 = 1, e_0 = 1,$
$s_{-1} = s_0 = 0, c_{-1} = c_0 = -1, \zeta_{-1} = \zeta_0 = 1$
for $k = 1, 2, \ldots$
    $\alpha_k = v_k^T H v_k$
    $u = Hv_k - \alpha_k v_k - \beta_k v_{k-1}$
    $\beta_{k+1} = |u|$
    $v_{k+1} = u/\beta_{k+1}$
    $\varepsilon_k = s_{k-2}\beta_k$
    $\bar{\delta}_k = -c_{k-2}\beta_k$
    $\delta_k = c_{k-1}\bar{\delta}_k + s_{k-1}\alpha_k$
    $\bar{\gamma}_k = s_{k-1}\bar{\delta}_k - c_{k-1}\alpha_k$
    $\gamma_k = \left(\bar{\gamma}_k^2 + \beta_{k+1}^2\right)^{1/2}$
    $c_k = \bar{\gamma}_k/\gamma_k$
    $s_k = \beta_{k+1}/\gamma_k$
    $\zeta_k = -(\varepsilon_k\zeta_{k-2} + \delta_k\zeta_{k-1})/\gamma_k$
    $\bar{w}_k = s_{k-1}\bar{w}_{k-1} - c_{k-1}v_k$
    $e_k = e_{k-1}|c_{k-1}s_k/c_k|$
    if $e_k \le \eta$
        $p^* = p_{k-1} + \zeta_k\bar{w}_k/c_k$
        stop
    endif
    $w_k = c_k\bar{w}_k + s_k v_{k+1}$
    $p_k = p_{k-1} + \zeta_k w_k$
    $k = k + 1$
end for
```

APPENDIX B: ALGORITHM OF THE DIMER METHOD

The dimer method evolves an image to a saddle point following the direction of the eigenvector corresponding to the lowest eigenvalue of the Hessian. At each iteration, the lowest eigenvalue and the corresponding eigenvector are first estimated by rotation of a dimer centered at the current iterate, then the system is translated based on a modified potential force. The algorithms given below are from Refs. 20 and 21.

Denote the center of the dimer by x and its orientation by $\hat{N}$. $F = -\nabla V(x)$ is the potential force at x. The algorithm below computes the lowest eigenvalue of the Hessian at x and the corresponding eigenvector.

Algorithm B1 (Rotation):

```
for $k = 1, 2, \ldots$
    $x_1 = x + \delta x_D \hat{N}$
    $F_1 = -\nabla V(x_1)$
    $F_2 = 2F - F_1$
    $F_{12} = F_1 - F_2$
    $F^\perp = (F_{12} - (F_{12}\cdot\hat{N})\hat{N})/\delta x_D$
    if $|F^\perp| < \delta F_D$
        $\lambda = -F_{12}\cdot\hat{N}/(2\delta x_D)$
        stop
    endif
    $\hat{\Theta} = F^\perp/|F^\perp|$
    $\hat{N}^* = (\cos\delta\theta_D)\hat{N} + (\sin\delta\theta_D)\hat{\Theta}$
    $\hat{\Theta}^* = -(\sin\delta\theta_D)\hat{N} + (\cos\delta\theta_D)\hat{\Theta}$
    $x_1^* = x + \delta x_D\hat{N}^*$
    $F_1^* = -\nabla V(x_1^*)$
    $F_2^* = 2F - F_1^*$
    $F_{12}^* = F_1^* - F_2^*$
    $f^{*\perp} = F_{12}^*\cdot\hat{\Theta}^*/\delta x_D$
    $f_r = (f^{*\perp} + |F^\perp|)/2$
    $c = (f^{*\perp} - |F^\perp|)/\delta\theta_D$
    $\Delta\theta = -(\arctan(2f_r/c) + \delta\theta_D)/2$
    if $c > 0$
        $\Delta\theta := \Delta\theta + \pi/2$
    endif
    $\hat{N} = (\cos\Delta\theta)\hat{N}^* + (\sin\Delta\theta)\hat{\Theta}^*$
    $\hat{N} := \hat{N}/|\hat{N}|$
    if $k = n_D^{max}$
        $\lambda^* = -F_{12}^*\cdot\hat{N}^*/(2\delta x_D)$
        $F^{*\perp} = (F_{12}^* - (F_{12}^*\cdot\hat{N}^*)\hat{N}^*)/\delta x_D$
        $\lambda = \lambda^* - \frac{1}{2}|F^{*\perp}|\tan(\Delta\theta - \delta\theta_D/2)$
        stop
    endif
    k:=k+1
end for
```

After the lowest eigenvalue λ and the corresponding eigenvector $\hat{N}$ are computed, the dimer is translated by a modified potential force in which the component of the potential force in the direction $\hat{N}$ is reversed. In this paper, we implemented two algorithms for the translation: One translates the dimer in the direction of the modified potential force (Algorithm B2), and the other translates the dimer in the conjugate gradient direction (Algorithm B3). As before, F denotes the potential force at the center of the dimer x.

Algorithm B2 (Translation):

```
if $\lambda > 0$
    $\tilde{N} = -\mathrm{sign}(F \cdot \hat{N})\hat{N}$
    $x := x + \Delta x^{max} \tilde{N}$
else
    $\tilde{F} = F - 2(F \cdot \hat{N})\hat{N}$
    $\tilde{N} = \tilde{F}/|\tilde{F}|$
    $x^* = x + \delta x_{lm} \tilde{N}$
    $F^* = -\nabla V(x^*)$
    $\tilde{F}^* = F^* - 2(F^* \cdot \hat{N})\hat{N}$
    $f_t = (\tilde{F}^* + \tilde{F}) \cdot \tilde{N}/2$
    $c = (\tilde{F}^* - \tilde{F}) \cdot \tilde{N}/\delta x_{lm}$
    $\Delta x = -f_t/c + \delta x_{lm}/2$
    $\Delta x := \mathrm{sign}(\Delta x) \min(\Delta x^{max}, |\Delta x|)$
    $x := x + \Delta x \tilde{N}$
endif
```

Algorithm B3 (Translation CG):

```
if $\lambda > 0$
    $\tilde{F} = -(F \cdot \hat{N})\hat{N}$
else
    $\tilde{F} = F - 2(F \cdot \hat{N})\hat{N}$
endif
$\gamma = (\tilde{F} - \tilde{F}^{old}) \cdot \tilde{F}/|\tilde{F}|^2$
$G = \tilde{F} + \gamma G^{old}$
$\tilde{N} = G/|G|$
if $\lambda > 0$
    $x := x + \Delta x^{max} \tilde{N}$
else
    $x^* = x + \delta x_{lm} \tilde{N}$
    $F^* = -\nabla V(x^*)$
    $\tilde{F}^* = F^* - 2(F^* \cdot \hat{N})\hat{N}$
    $f_t = (\tilde{F}^* + \tilde{F}) \cdot \tilde{N}/2$
    $c = (\tilde{F}^* - \tilde{F}) \cdot \tilde{N}/\delta x_{lm}$
    $\Delta x = -f_t/c + \delta x_{lm}/2$
    $\Delta x := \mathrm{sign}(\Delta x) \min(\Delta x^{max}, |\Delta x|)$
    $x := x + \Delta x \tilde{N}$
endif
```

In the above algorithm, $\tilde{F}^{old}$ and $\tilde{G}^{old}$ are the modified force and the conjugate gradient direction in the previous iteration, respectively.

The rotation and translation steps are repeated until the maximum norm of the potential force at x falls below δ_n. There are a number of parameters whose values need to be specified in the algorithm. In the numerical examples, we used the values suggested in Ref. 21: $\delta x_D = 10^{-4}$, $\delta F_D = 0.1$, $\delta\theta_D = 10^{-4}$, $n_D^{max} = 1$, $\Delta x^{max} = 0.1$, $\delta x_{lm} = 10^{-3}$.

[1] R. Elber and M. Karplus, Chem. Phys. Lett. **139**, 375 (1987).
[2] R. Czerminski and R. Elber, Int. J. Quantum Chem. **38**, 167 (1990).
[3] S. Fischer and M. Karplus, Chem. Phys. Lett. **194**, 252 (1992).
[4] I. V. Ionova and E. A. Carter, J. Chem. Phys. **98**, 6377 (1993).
[5] H. Jónsson, G. Mills, and K. W. Jacobsen, "Nudged elastic band method for finding minimum energy paths of transitions," in *Classical and Quantum Dynamics in Condensed Phase Simulations*, edited by B. J. Berne, G. Ciccoti, and D. F. Coker (World Scientific, 1998).
[6] G. Henkelman, B. P. Uberuaga, and H. Jónsson, J. Chem. Phys. **113**, 9901 (2000).
[7] W. E, W. Ren, and E. Vanden-Eijnden, Phys. Rev. B **66**, 052301 (2002).
[8] W. E, W. Ren, and E. Vanden-Eijnden, J. Chem. Phys. **126**, 164103 (2007).
[9] G. M. Crippen and H. A. Scheraga, Arch. Biochem. Biophys. **144**, 462 (1971).
[10] C. J. Cerjan and W. H. Miller, J. Chem. Phys. **75**, 2800 (1981).
[11] J. Simons, P. Jorgensen, H. Taylor, and J. Ozment, J. Phys. Chem. **87**, 2745 (1983).
[12] S. Bell and J. S. Crighton, J. Chem. Phys. **80**, 2464 (1984).
[13] A. Banerjee, N. Adams, J. Simons, and R. Shepard, J. Phys. Chem. **89**, 52 (1985).
[14] J. Baker, J. Comput. Chem. **7**, 385 (1986).
[15] P. Y. Ayala and H. B. Schlegel, J. Chem. Phys. **107**, 375 (1997).
[16] L. J. Munro and D. J. Wales, Phys. Rev. B **59**, 3969 (1999).
[17] Y. Kumeda, D. J. Wales, and L. J. Munro, Chem. Phys. Lett. **341**, 185 (2001).
[18] N. Mousseau and G. T. Barkema, Phys. Rev. E **57**, 2419 (1998).
[19] E. Cancès, F. Legoll, M.-C. Marinica, K. Minoukadeh, and F. Willaime, J. Chem. Phys. **130**, 114711 (2009).
[20] G. Henkelman and H. Jónsson, J. Chem. Phys. **111**, 7010 (1999).
[21] R. A. Olsen, G. J. Kroes, G. Henkelman, A. Arnaldsson, and H. Jónsson, J. Chem. Phys. **121**, 9776 (2004).
[22] J. Klimeš, D. R. Bowler, and A. Michaelides, J. Phys.: Condens. Matter **22**, 074203 (2010).
[23] W. E and X. Zhou, Nonlinearity **24**, 1831 (2011).
[24] J. Zhang and Q. Du, SIAM J. Numer. Anal. **50**, 1899 (2012).
[25] J. Zhang and Q. Du, J. Comput. Phys. **231**, 4745 (2012).
[26] A. F. Voter, Phys. Rev. Lett. **78**, 3908 (1997).
[27] A. F. Voter, J. Chem. Phys. **106**, 4665 (1997).
[28] A. F. Voter, Phys. Rev. B **57**, R13985 (1998).
[29] M. R. Sørensen and A. F. Voter, J. Chem. Phys. **112**, 9599 (2000).
[30] C. C. Paige and M. A. Saunders, SIAM J. Numer. Anal. **12**, 617 (1975).
[31] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series on Operations Research (Springer, 1999).