

**A GLOBAL SEARCH ALGORITHM FOR PHASE TRANSITION
PATHWAYS IN COMPUTER-AIDED NANO-DESIGN**

A thesis
Presented to
The Academic Faculty

by

Lijuan He

In Partial Fulfillment
of the Requirement for the Degree
Master of Science in the
School of Mechanical Engineering

Georgia Institute of Technology
December 2013

COPYRIGHT 2013 BY LIJUAN HE

**A GLOBAL SEARCH ALGORITHM FOR PHASE TRANSITION
PATHWAYS IN COMPUTER-AIDED NANO-DESIGN**

Approved by

Dr. Yan Wang, Advisor

School of Mechanical Engineering

Georgia Institute of Technology

Dr. Ting Zhu

School of Mechanical Engineering

Georgia Institute of Technology

Dr. Seung Soon Jang

School of Material Science & Engineering

Georgia Institute of Technology

Date Approved: July 19, 2013

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Yan Wang, for his guidance, support and patience during the past two years study. My research is always inspired by talking with him. It is his encouragement and feedback in my efforts that make this thesis possible.

I would also like to thank Dr. Ting Zhu and Dr. Seung Soon Jang for serving on my committee. Taking their courses, greatly benefits the finish of the thesis by giving me substantial background knowledge grounding this research.

I am grateful to every other student in Dr. Wang's research group for their advice and friendship. Special thanks go to Edin Crnkic, who provided useful advice on my thesis. I would like to thank my parents for encouraging me whenever I came into frustrated.

I would like to acknowledge the support of the National Science Foundation grant CMMI-1001040, which provided me with the financial support for the research done in this thesis.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	II
LIST OF TABLES	V
LIST OF FIGURES	VI
SUMMARY	VII
CHAPTER 1 INTRODUCTION.....	1
1.1 MOTIVATION.....	1
1.2 AN OVERVIEW OF THE PROPOSED GLOBAL SADDLE-POINT SEARCH METHOD	4
1.3 CONTRIBUTIONS.....	5
CHAPTER 2 BACKGROUND.....	7
2.1 EXISTING TRANSITION PATHWAY SEARCH METHODS.....	8
2.2 EXISTING SADDLE POINT SEARCH METHODS.....	13
2.3 CONJUGATE GRADIENT METHOD.....	15
2.4 EXISTING METHODS FOR DEGREE REDUCTION OF BÉZIER CURVE	17
CHAPTER 3 CONCURRENT MULTI TRANSITION PATHWAY SEARCH ALGORITHM	20
3.1 A SINGLE TRANSITION PATHWAY SEARCH	21
3.1.1 Searching the Stable Configuration	22
3.1.2 Searching the MEP.....	25
3.1.3 Constrained Degree Elevation and Degree Reduction.....	26
3.1.4 Local Degree Elevation and Degree Reduction.....	31
3.2 MULTIPLE TRANSITION PATHWAY SEARCH	31
3.2.1 Scheme for Selecting Breakpoint.....	34
3.2.2 Discussion on the subdivision scheme.....	42
CHAPTER 4 IMPLEMENTATION AND DEMONSTRATION	43
4.1 TEST RESULT FOR LEPS POTENTIAL	45
4.2 TEST RESULT FOR LEPS PLUS HARMONIC OSCILLATOR POTENTIAL.....	46

4.3	TEST RESULT FOR RASTRIGIN FUNCTION	48
4.4	TEST RESULT FOR SCHWEFEL FUNCTION.....	51
4.5	DISCUSSION	54
4.5.1	Convergence analysis.....	54
CHAPTER 5 SUMMARY AND FUTURE WORK		56
5.1	SUMMARY AND DISCUSSIONS	56
5.2	FUTURE WORK	58
APPENDIX.....		61
REFERENCES.....		95

LIST OF TABLES

Table 1: Pseudo-code of the algorithm for a single transition pathway search	24
Table 2: Pseudo-code of the algorithm for multiple transition pathway search	33
Table 3: Curve subdivision scheme (five control points)	37
Table 4: Curve subdivision scheme (six control points).....	38
Table 5: Pseudo-code of the curve division scheme (five and six control points)	39
Table 6: Test functions.....	44
Table 7: Test results on Rastrigin function (contour plot refer Figure 7).....	49
Table 8: Test results on Rastrigin function (contour plot refer Figure 8).....	50
Table 9: Test results on Schwefel function (contour plot refer Fig. 9).....	53
Table 10: Test results on Schwefel function (contour plot refer Fig. 10).....	54

LIST OF FIGURES

Figure 1: Flow chart for the procedure of the algorithm	21
Figure 2: Flow chart for the procedure of a single transition path search	23
Fig. 3: Illustration for multiple pathway search (five control points).....	35
Fig. 4: Illustration for multiple pathway search (six control points)	37
Fig. 5: Test results for LEPS potential function	47
Fig. 6: Test results for LEPS plus harmonic oscillator potential function.....	48
Fig. 7: Test result for Rastrigin function with the initial position at (-2.81, 0.50), (-1.43, 2.90), (0.23, -2.47), (1.57, 2.67), and (2.91, -0.11).	50
Fig. 8: Test result for Rastrigin function with the initial position at (-2.81, -1.50), (-1.43, -1.50), (0.23, -1.50), (1.57, -1.50), and (2.91, -1.50).....	51
Fig. 9: Test result for Schwefel function with the initial position at (-100.3, 25), (-40.5, -45), (17.8, 50.3), (69.8, 70.6), and (130.2, 98.7).....	52
Fig. 10: Test result for Schwefel function with the initial position at (-100.3, 25), (-40.5, 25), (17.8, 25), (69.8, 25), and (130.2, 25).....	53
Fig. 11: Illustration for two possible transition paths between two states	59

SUMMARY

One of the most important design issues for phase change materials is to engineer the phase transition process. The challenge of accurately predicting a phase transition is estimating the true value of transition rate, which is determined by the saddle point with the minimum energy barrier between stable states on the potential energy surface (PES). In this thesis, a new algorithm for searching the minimum energy path (MEP) is presented. The new algorithm is able to locate both the saddle point and local minima simultaneously. Therefore no prior knowledge of the precise positions for the reactant and product on the PES is needed. Unlike existing pathway search methods, the algorithm is able to search multiple transition paths on the PES simultaneously, which gives us a more comprehensive view of the energy landscape than searching individual ones. In this method, a Bézier curve is used to represent each transition path. During the searching process, the reactant and product states are located by minimizing the two end control points of the curve, while the shape of the transition pathway is refined by moving the intermediate control points of the curve in the conjugate directions. A curve subdivision scheme is developed so that multiple transitions paths can be located. The algorithm is demonstrated by examples of LEPS potential, LEPS plus harmonic oscillator potential, and PESs defined by Rastrigin function and Schwefel function.

CHAPTER 1

INTRODUCTION

In order to speed up the revolution of nanotechnology, we should adopt a focus shift from the ‘discovery-based’ development approach to a ‘solution-oriented’ approach to create new nanomaterials. In the discovery-based approach, new materials are discovered first and then their potential applications are explored. In the solution-oriented approach, new materials are *designed* systematically to solve existing problems. Designing new materials with the aim to solve engineering problems will be a major mission for engineers in the future. The availability of computational design tools is the key to improve the efficiency of the materials design process. The research presented in this thesis is to provide such kind of design tools, specifically the ones for simulation-based phase change materials design. In the rest of this chapter, the motivation of our research and the problem we try to solve are first described. A general description of the proposed method to solve the problem is given, followed by a summary of the contributions of this thesis.

1.1 Motivation

Nanotechnology of today mainly focuses on the discovery of nanoscale materials with new properties and applications. So far it heavily relies on trial-and-error laboratory experiments to turn them into commercial products for potential applications. This discovery-based development approach is a time-consuming process and will not be able to satisfy the increasing demand on new materials in the future. A solution-oriented approach should be adopted instead. That is, engineers start with the analysis of the existing needs in solving engineering problems, then design and produce new materials that can fulfill those needs in order to solve the specific problems. Engineers need enabling technologies to create novel materials by designing the microstructure of the materials systematically through a bottom-up approach. In order to achieve this goal, computational design tools are essential. For example, modeling and simulation software allows engineers to accurately predict the properties and functions of different material structures to improve the efficiency of the materials design process.

Among engineering materials, phase change materials have attracted much attention and

have been widely used in engineering products such as in information storage (e.g. hard-disk, CD-ROM, memory) and in energy storage (e.g. battery, shape memory alloy). One of the most important design issues for these materials is to engineer the phase transition process. Simulation that predicts the transition processes efficiently and accurately under desirable conditions is critical in the design of phase change materials. The ultimate goal for the research presented in this thesis is to provide simulation tools to design phase transition processes.

Traditionally, a phase transition is described from a top-down viewpoint as the transformation of a thermodynamic system from one phase to another. A phase is one of the states of matter which has uniform physical properties, and the materials system has a particular level of free energy. When external conditions are varied, such as an enough change in temperature or pressure, one or more properties of the material change and a phase transition occurs. The system shifts from one free energy level to another as a result of these external influences. The external conditions and amount of required input energy can be quantitatively estimated and are the key to define a phase transition. For example, water has three phases (gas, liquid, and solid) from the thermodynamic viewpoint. A liquid water changes to gas water when it is boiled. In other words, a change in temperature can result in a phase transition from liquid to gas. In general, a phase transition is accompanied by a series of physical events. There are two broad classifications of phase transition, one involved latent heat and the other does not. Also there are some other classifications such as the categories of first order, second order, and infinite order phase transition. More details about the phase transition in thermodynamic point of view can be found in [1] and [2].

A phase transition process can also be described from a bottom-up viewpoint, which is adopted in this thesis. From the material's microstructure point of view, a phase transition is a geometric and topological transformation process of materials from one phase to another, each of which has a unique and homogeneous physical property. In this description, we are interested in the structure and topology changes for the material at atomistic scale. Understanding and controlling the phase transition from the microscopic level is critical to design phase change materials. In simulation-based design, the challenge of accurately predicting a phase transition is the knowledge of the transition rate, which is determined by the energy barrier that exists between the initial and final states. That is, activation energy is required to enable the transition

from the initial structure to the final one. Mathematically it is the saddle point on the potential energy surface (PES) with the minimum energy barrier that determines the transition rate. The general process to simulate the phase transition process is as follows. First, a PES which characterizes detailed information about energy landscape is generated. Then a minimum energy path (MEP) which is the most probable physical pathway of transition among all possible ones is located. Finally the activation energy is obtained by finding the maximum energy on the MEP and the transition rate is calculated using the transition state theory [3]. The key of phase transition simulation is searching the saddle points on the PES.

Besides designing phase change materials, phase transition simulation is also important in other applications. For example, the diffusion of adatoms on a solid, which is a fundamental problem in surface science and has attracted attentions for decades, can be regarded as a phase transition process. Chemical reactions and protein folding can also be generally treated as phase transitions. Therefore, simulating phase transitions is also essential to design physical and chemical processes at atomistic scales.

Compared to vibrations and other thermal behaviors, phase transitions are rare events. Kinetic Monte Carlo (KMC) [4] simulation has been widely used to simulate the rare events, which are characterized by occasional transitions from one state to another, with long periods of relatively inactivity between these transitions. Traditional molecular dynamics (MD) simulation focuses on femto-second trajectory prediction and is not efficient in simulating rare events. KMC has been extensively used in physics and chemistry fields, such as the process of surface diffusion, vacancy diffusion in alloys, etc. One of the key inputs for KMC simulation is the rate for each of the possible events. The process of the surface diffusion in KMC consists of a sequence of atom jumping events. The configuration of the system changes after each event. All the possible configurations are called the states of the system, each of which corresponds to a specific energy level. The transition rate from one state to another is determined by the energy barrier between the two states.

Many numerical methods have been developed to search MEP on the PES, such as the nudged elastic band (NEB) method [5] and the string method [6]. These existing transition pathway search methods only search one path locally. The initial or final states have to be known a priori. The purpose of this research is to develop a new method to locate multiple saddle points

without knowing the reactant and product in advance. It provides a global view of the energy landscape. Thus the accuracy and efficiency of prediction can be improved.

1.2 An Overview of the Proposed Global Saddle-Point Search Method

A new algorithm for searching the MEP is presented in this thesis. The new algorithm is able to locate both the saddle point and local minima simultaneously. Therefore no prior knowledge of the precise positions for the reactant and product on the PES is needed. Unlike existing pathway search methods, the algorithm is able to search multiple transition paths on the PES simultaneously, which can provide a global view of the energy landscape, instead of a local one by searching individual paths.

Here the Bézier curve is used to represent the transition pathway. A Bézier curve is a parametric curve defined by control points which can be used to manipulate the shape of the curve. In our global search algorithm, each control point of the curve represents one state, also called image, on the transition path. During the searching process, reactant and product states are located by minimizing the two end control points of the Bézier curve using the conjugate gradient method, while the shape of the transition pathway is refined by moving the intermediate control points of the curve in the conjugate directions. In each iteration, a set of conjugate directions are determined and then several line minimization steps along those conjugate directions are applied to each intermediate control points. As a result, the Bézier curve will gradually converge to the MEP. In order to keep the control points evenly distributed, a modified Bézier curve degree elevation and reduction scheme is developed here to redistribute the control points in each iteration. Since there could be more than one saddle point with extra local minima between the two stable states, one curve could be broken into two to represent two stages of transitions recursively. We also developed a curve subdivision scheme to check whether there is more than one saddle point with extra local minima between the two end points of the optimized curve. If there is, we break this curve into two which are regarded as the initial guess of the transition path for the two stages of transitions. Then those two curves are optimized by the same procedure as we did to the original curve. This ‘check-and-break’ process continues until each of the curves only passes through two adjacent local minimums with their end points located at those local minimums. Since in real applications, people are more concerned about the exact positions of the saddle points instead of the MEP in most cases, a climbing up scheme is also

introduced to locate the saddle points on those curves that are close enough to the MEP.

1.3 Contributions

The new algorithm is able to search multiple transition paths on the PES simultaneously, which can provide a global view of the energy landscape. The existing methods only search one transition path connecting the reactant and product with an implicit assumption of one transition stage. For materials with complicated potential energy surfaces, often there are more than one transition stages with several metastable states between the two stable configurations. There is a possibility that the existing methods miss the saddle point with the highest energy level, which could lead to an underestimation of the energy barrier between two states. For example, for a process during which the reactant needs to go through several other stable states (local minima) to reach the product, the NEB method may end up with locating some saddle points which are not necessary the one with highest energy since the number of images for NEB method is fixed during searching process, The proposed method can solve this problem by locating all the metastable states as well as the corresponding saddle points along the path between the reactant and product. Once all the metastable states and saddle points are located, it is easy to estimate the minimum energy barrier between the reactant and product. In addition, it is a more efficient way to search multiple transition paths simultaneously than searching individual ones in the situation when a comprehensive view of energy landscape is important. For the existing searching methods, it is difficult to receive a complete view of the distribution of the local minimums and saddle points on the energy surface. It may need a significant number of trials with different initial guesses in order to obtain a good sense of the energy landscape. The proposed method can locate several local minimums and saddle points for a given range at a time. As a result, it can give us a global view of the energy landscape.

The proposed method shares some similarities with both the NEB and string methods in terms of treating the transition process as discretized images, and using several points or nodes to represent those images. Different from the NEB and string methods, the proposed method uses a Bézier curve to represent the transition path. The control points of the Bézier curve represent the discretized images on the transition path. By moving the control points, the curve will gradually converge to the MEP. For the NEB method, it employs spring forces to keep the nodes uniformly spaced along the path. As a result, it needs to determine the Hooke constants for springs as well

as tangent directions onto which the spring forces will project in each iteration. The spline NEB and string methods adopts re-parameterization scheme to keep the images on the path relatively evenly distributed. Different from them, our method use the degree elevation and reduction scheme of the Bézier curve to keep the control points well distributed. Since the degree elevation and reduction of the Bézier curve use the simple linear interpolation of the control points, the computation cost is much less than NEB and string methods for the redistribution process.

The rest of the thesis is organized as follows. Chapter 2 provides some background knowledge related to the topics of transition pathway search, saddle point search, convergence analysis of the conjugate gradient method, and the Bézier curve degree reduction methods. Chapter 3 gives a detailed description of our new global search method. Chapter 4 shows the implementation of our method and the demonstration with examples. Chapter 5 concludes with a brief summary of the accomplishment and discussions about the future extension.

CHAPTER 2

BACKGROUND

This chapter presents some background knowledge for our global search algorithm. The challenge for phase transition simulation is to search the transition rate which is determined by the activation energy between two states. An activation energy barrier always exists between two states. In 1931, Eyring and Polanyi [3, 8] proposed the transition state theory (TST) as a means to calculate the transition rates using the activation energy to characterize reactions. Most of the simulation methods developed recently are based on the TST and harmonic transition state theory (hTST) [9]. Some variants of TST (Variational Transition State Theory [10] and Reaction Path Hamiltonian [11]) are also used. The general procedure to simulate a phase transition process is as follows. First a PES is generated. Then a MEP which is the most probable physical pathway of transition among all possible ones is located. Finally the activation energy is obtained by finding the maximum energy on the MEP and the transition rate is calculated using TST. Subsequently the phase transition simulation can be done using KMC, Accelerated Molecular Dynamics (AMD) [12] or other simulation methods.

The accuracy of the simulation depends on the accuracy of the rate constants. In other words, it depends on the accuracy of the activation energy. The research on transition pathway search and saddle point search aims to find the accurate MEP and the saddle point. There are two challenges in locating the MEP and the saddle point. One is the generation of an accurate PES; the other is to search the MEP and saddle point on the generated PES.

Tremendous research has been done on how to generate the PES and many methods were developed. Reference [13] presents a review of available methods of generating the PES characterizing information regarding the interatomic and intermolecular interactions that characterize the reaction is included in reference. [14-19] give some examples of PES generation methods. Libraries and repositories of PES are also available and ready for use [20]. Further discussion of these PES generation methods is beyond the scope of this thesis.

The search process of MEP and saddle points is also intensively studied. Many different methods were developed to find the transition path and saddle points. [21-25] are examples of the available reviews that give a detailed review or even comparison on some of those methods.

In general, there are two types of scheme for the searching process. One is the single-ended search and the other is double-ended search. The scheme for the single-ended search methods starts at a local minimum on the PES which represents the initial state configuration, then traces stepwise sequentially until the MEP and saddle points are located. The activation-relaxation method [26] and dimer method [27] are the examples of single-ended search methods. The double-ended search methods make use of the two minima on the PES. Those methods require the information for both the initial and final stable configurations. Examples includes the DHS method [28], the ridge method [29] and the NEB method [5], etc. Sections 2.1 and 2.2 give a detailed review on the existing reaction pathway search and saddle points search methods respectively.

As mentioned in section 1.2, the conjugate gradient method is employed in our method to locate the local minimums. A brief introduction and a literature review of the convergence analysis on this method are presented in section 2.3. In section 2.4, a review of the existing methods on the Bézier curve degree reduction is presented.

2.1 *Existing transition pathway search methods*

Since 1970, there have been many methods developed to search the stationary points. The methods in Refs. [30-32] are some examples, which are not going to be reviewed in this thesis. Instead, we focus on more recent ones which aim to search the MEP on the PES.

Basically, the transition pathway search methods can be classified either as chain-of-states methods, including NEB and string methods, or as one of the other methods. Chain-of-states methods rely on a collection of images that represent intermediate states of the atomic structure as it transforms from the initial to the final configuration along the transition path. After an optimization on all the intermediate images simultaneously, these discrete states are chained to each other when the search converges, usually by interpolating between the states, and the transition pathway and saddle point are obtained. They work well in transitions where there may be more than one saddle point, i.e. there may be more than one transition state. In situations where there may be multiple transition pathways, the methods will converge to the pathway which is closest to the initial guess of the transition pathway.

The early version of the chain of state methods is the plain elastic band (PEB) method

[33-39], which represents the transition path by several discrete images normally in a range of four to twenty images. These images are subject to the total potential force and spring force. After several rounds of optimization, the forces on those images vanish and the transition path converges to the MEP. But in practice, the PEB method fails to locate the MEP in most situations. The PEB method has two major defects, which are “corner cutting” and “sliding down”. The corner cutting issue is caused by large spring stiffness. The band is too stiff and the path cuts the corner which leads to overestimating the saddle point energy. The sliding down issue occurs when the stiffness of the spring is too small. The images slide down when getting closer to the saddle point region and avoid the barrier region, thus reducing the resolution of the path in the most critical area. These two defects are the motivation for developing the NEB [5] method, one of the most commonly used chain-of-state methods. The NEB method represents the transition path by a set of images connected by spring in order to ensure the continuity of the path. An essential feature of the NEB method, which distinguishes it from other chain of state methods, is the projection of the forces which ensures that the spring forces only control the distribution of the images, and the true force only controls the convergence of the band to the MEP. In each iteration, the tangent direction to the path should be determined to decompose the true force and the spring force into parallel and perpendicular components. The perpendicular components of the spring force and the parallel components of the true force are eliminated. This force projection is referred as nudging. As a result, the spring force can be varied by several orders of magnitude without introducing the ‘corner cutting’ or ‘sliding down’ problems. The resolution at the region of interest (ROI) and the accuracy of the saddle point energy estimation are improved. The force on image i then becomes

$$\mathbf{F}_i = -\nabla V(\mathbf{R}_i)|_{\perp} + \mathbf{F}_i^s \cdot \hat{\boldsymbol{\tau}}_{\parallel} \hat{\boldsymbol{\tau}}_{\parallel} \quad (2.1)$$

where \mathbf{R}_i is the position vector for the i^{th} image; $\nabla V(\mathbf{R}_i)|_{\perp}$ indicates the perpendicular component of the potential force; \mathbf{F}_i^s represents the total spring force on the i^{th} image; and $\hat{\boldsymbol{\tau}}_{\parallel}$ is the unit tangent to the path. In each iteration, the tangent vector along the path is defined by central finite difference approximation:

$$\boldsymbol{\tau}_i = \frac{\mathbf{R}_i - \mathbf{R}_{i-1}}{|\mathbf{R}_i - \mathbf{R}_{i-1}|} + \frac{\mathbf{R}_{i+1} - \mathbf{R}_i}{|\mathbf{R}_{i+1} - \mathbf{R}_i|} \quad (2.2)$$

The unit tangent vector $\hat{\boldsymbol{\tau}}_{\parallel}$ is obtained by normalizing the tangent vector $\boldsymbol{\tau}_i$. The NEB method requires the knowledge of the initial and final states of the transition, as well as an initial guess of the transition path. At each iteration, the forces acting on the intermediate images are minimized using an optimization algorithm, e.g. the Broyden-Fletcher-Goldfarb-Shanno (BFGS method), keeping the initial and final states fixed. As a result, the images iteratively converge to the MEP. To interpret the results, the interpolation between adjacent images should be adopted to get the MEP. In the event of multiple MEPs, the algorithm will converge to the MEP which is closest to the initial guess of the path. The algorithm works well in most situations. But in the systems where the force along the MEP is larger compared to the restoring force perpendicular to the path, kinks may develop because no perpendicular spring forces are considered. As a result, it prevents the band from converging to the MEP. In addition, in some cases, the actual saddle point may not locate at the position at one of the intermediate images. The improved tangent NEB [40] and doubly nudged elastic band (DNEB) [41] methods reduce the appearance of kinks by generating a better estimation of the tangent direction of the path and re-introducing a perpendicular spring force component, respectively. In the improved tangent method, instead of using both the adjacent images \mathbf{R}_{i-1} and \mathbf{R}_{i+1} , only the image with higher energy and the image \mathbf{R}_i itself are used to estimate the tangent direction. The new definition of the tangent direction is

$$\boldsymbol{\tau}_i = \begin{cases} \mathbf{R}_{i+1} - \mathbf{R}_i & \text{if } V_{i+1} > V_i > V_{i-1} \\ \mathbf{R}_i - \mathbf{R}_{i-1} & \text{if } V_{i+1} > V_i > V_{i-1} \end{cases} \quad (2.3)$$

when the image \mathbf{R}_i is at a maximum or a minimum in energy, the tangent vector is defined as a weighted average of the vectors to the two neighboring images. The weight is determined based on the energy. In the DNEB method, a manipulated perpendicular component of the spring force is added back to the total force in order to reduce kinks. The total force then becomes

$$\mathbf{F}_i = -\nabla V(\mathbf{R}_i)|_{\perp} + \mathbf{F}_i^s|_{\parallel} + \mathbf{F}_i^{s*} \quad (2.4)$$

where

$$\mathbf{F}_i^{s*} = \mathbf{F}_i^s \Big|_{\perp} - (\mathbf{F}_i^s \Big|_{\perp}) \cdot \hat{\boldsymbol{\tau}}_i \hat{\boldsymbol{\tau}}_i \quad (2.5)$$

The band is now doubly nudged as a result of inclusion both components of the spring force. During the force minimization process, the limited-memory quasi-Newton (L-BFGS) [42] optimization method is used, which accelerates the convergence of the relaxation process. The climbing image NEB (CI-NEB) [43] is a modification to the NEB method. This method not only retains the shape information of the MEP but also allows one image converges to the saddle point rigorously. After a few iterations of regular NEB, the image i_{\max} with the highest energy is identified. Only for i_{\max} , the force is calculated separately as:

$$\mathbf{F}_{i_{\max}} = -\nabla V(\mathbf{R}_{i_{\max}}) + 2\nabla V(\mathbf{R}_{i_{\max}}) \cdot \hat{\boldsymbol{\tau}}_{i_{\max}} \hat{\boldsymbol{\tau}}_{i_{\max}} \quad (2.6)$$

This is the total potential force with the component along the band being inverted. No spring force is applied to image i_{\max} , which allows the image i_{\max} actively climbing towards the saddle point along the band. The forces on other images are defined in the same way as it does in the NEB method. The only difference is that the spring constants are calculated as a function of the maximum energy along the band, which leads to a maximum spring constant if the image locates at the maximum energy position and a small spring constant if the images are away from its maximum energy position. As a result, there are more images settling around the saddle point and thus achieving higher resolution at this critical region. The free end CINEB [44] method achieves an important improvement to the original CI-NEB method by allowing one end state swinging freely at a given level of energy. The movable end state is subject to a force defined as

$$\mathbf{F}_p = -\frac{\left(\mathbf{F}_p^s \Big|_{\parallel} \cdot \nabla V(\mathbf{R}_p)\right) \nabla V(\mathbf{R}_p)}{\|\nabla V(\mathbf{R}_p)\|^2} + \mathbf{F}_p^s \Big|_{\parallel} \quad (2.7)$$

In such a way, one can keep the moving end state close to the saddle point region. As a result, the number of images along the band can be significantly reduced while retaining a reasonable resolution around the saddle point region. The computational cost to locate the saddle point is greatly reduced. The spline NEB [45] method has two modifications from the original NEB method. One is using a second-order L-BFGS in the relaxation process of each intermediate image along the band, which leads to a faster convergence to the MEP. The other one is using a

spline interpolation to represent the path while eliminating the spring forces between adjacent images along the band, which makes the method more robust. After the band is converged, the Eigenvector Following [46, 47] optimization method can be applied further to locate actual saddle points, and the resolution of ROI can be increased by using adaptive spring constants.

String method [6, 48] is another commonly used chain of state method. The string method uses a smooth curve with intrinsic parameterization such as arclength, or energy weighted arclength to represent the transition pathway. The string can be discretized into number of points called the images on the string. As opposed to NEB, the number of points used in the string method can be modified dynamically. The MEP is located by evolving the discretized images according to the perpendicular components of the potential force. The improved string method [49] evolves the string according to the full potential force instead of perpendicular component of the force. That is, the force projection step is eliminated, which makes the method more stable and accurate. The algorithm involves two simple steps: evolving the string by standard ordinary differential equation (ODE) solvers; and re-parameterization of the string by interpolation. The growing string method [50] does not require the initial guess of a complete transition path. It grows from the ends of reactant and product until these two end points join together and then the joint curve evolves to the MEP as the original string method does. This method consists of a two-step procedure: evolution and parameterization. In the evolution step, the total forces on the images are minimized. In the parameterization step, the images are redistributed along the string with a prescribed density. If the perpendicular force at a frontier node is less than a threshold, one more node is added and the node continues grow. The quadratic string [51] method is a variation of string method based on multi-objective optimization subject to constraints of parameterization and evolving direction. One essential modification made in this method is that the integration is done locally on a quadratic PES approximation. A damped Broyden-Fletcher-Goldfarb-Shanno (BFGS) Hessian update is applied in searching the MEP. The integration is performed with an adaptive step-size solver, which is restricted in length to the trust radius of the approximate Hessian. The step size in this method can be larger than the original string method.

Methods that are not classified as chain-of-states include the conjugate peak refinement (CPR) method [52], the accelerated Langevin dynamics (ALD) method [53], the concerted

variational strategy [54], and Hamilton-Jacobi method [55]. The CPR method iteratively finds a series of saddle points that are connected to each other and form a continuous reaction path from reactant to product by linear interpolation. It uses the fact that the Hessian matrix (second derivative of the energy surface) has exactly one negative eigenvalue and all others are positive at the first order saddle point. Based on this fact, the CPR method constructs a set of conjugate basis using the Beal's [56, 57] formula by setting the initial maximum direction as the direction along the line connecting the reactant and product. By using line maximization in one direction and line minimization along all other conjugate directions, one saddle point is located. This process is repeated to locate all other saddle points. The ALD method [53] is a stochastic transition path sampling method by solving the Langevin equation (LE) describing the stochastic dynamics of a thermally activated system. This method starts from the initial state and does not require the prior knowledge of the final state and saddle point. For a single transition, the pathway is divided into activation path (from initial state to one particular state M) which is generated by a deterministic Newtonian equation with negative friction and deactivation path (from state M to final state) which is generated by a regular Newtonian equation with positive friction. The transition path can be approximated as a weighted average of all the possible paths along the activation phase and the deactivation phase. The ALD method requires a priori knowledge of the activation time from the initial state to the state M in physical transitions. The concerted variational strategy [54] describes the transition path based on Maupertuis' and Hamilton's Principles. After obtaining the transition path, the conjugate residual method [58] for local search is used to locate the saddle point. The Hamilton-Jacobi method [55] relies on the solution of a Hamilton-Jacobi type equation to generate the MEP. The search is based entirely on the knowledge of the reactant. No prior knowledge for the product is needed. This method works on a cost principle: points with a higher potential energy level have a larger cost than points with a lower potential energy. A special cost is defined for the MEP.

2.2 Existing saddle point search methods

Instead of searching the complete MEP, saddle point search methods only locate the saddle point on the MEP. They are categorized into local and global search methods. One of the original local methods is the automated surface walking algorithm [59] which is based on eigenvectors of the Hessian matrix with local quadratic approximation of the PES within a trust

region. For each iteration, the trust radius and Hessian matrix are updated. The search process starts with tracking the eigenvalue of the updated Hessian matrix till the convergence is reached. The other one is the partitioned rational function optimization [60] method, which is based on local rational function approximation instead of quadratic approximation to the PES with augmented Hessian. The more recent ridge method [29] and dimer method [7] use a pair of images to search the saddle point without evaluating the Hessian matrix. The ridge method starts searching the maximum on the straight line connecting reactant and product. And then two images on the line with a small distance from the maximum are created. Those two images are then relaxed and a new maximum point along the straight line connecting the two new images is found. The process is repeated till the saddle point is located. The dimer method is a minimum-mode following saddle point search algorithm. It uses a pair of two images of the system, called dimer. The saddle point is located by minimizing the energy using rotation and translation at the center of the dimer iteratively. The improved dimer method [61] modifies the original dimer method by using a different way to calculate curvature, reducing the number of gradient calculations, choosing a larger step of rotation, and applying translation to the dimer middle instead of one of the two images. Those modifications significantly improve the overall performance and robustness of the algorithm on quantum-chemical PES which subjects to numerical noises. Reduced Gradient Following (RGF) [62] and Reduced Potential Energy Surface Model [63] methods use intersections of zero-gradient curves and surfaces, with saddle point search occurring within the subspace of these curves or surfaces. The RGF method involves predict and correct steps. It starts from a stationary point and then moves along the tangent of reduce gradient curves. The improved RGF method [64] makes the method more efficient by using an implied corrector step for each predictor, which reduces the number of calls to the computationally expensive corrector step. Finally, the Synchronous Transit method [65] locates the saddle point by using a single line minimization of energy along the direction orthogonal to the linear synchronous transit path, which is followed by the energy maximization along the quadratic synchronous transit path. [66] generalized and improved the method by estimating the transition state and refining the saddle point estimation through conjugate gradient optimization.

Local search methods may locate the saddle point which does not have the maximum energy on the MEP if there are multiple saddle points. Global search methods have the advantage

that the saddle point with the maximum energy is located if the search converges. The Dewar-Healy-Stewart method [28] searches for the saddle point by iteratively reducing the distance between reactant and product images. The main idea of this method is to pull the lower energy image over the potential energy surface towards the higher energy image. The Activation-Relaxation technique [26] can travel between many saddle points using a two-step process; an image first jumps from a local minimum to a saddle point, and then back down to another minimum. This method does not require an initial guess. The Step and Slide method [67] uses an image from the initial and final states. Energy levels of each are increased gradually, and the distance between them is minimized while remaining on the same isoenergy surface. The interval Newton's method [68] is capable of finding all stationary points by solving the equation of vanishing gradients.

2.3 *Conjugate gradient method*

Conjugate gradient method is an effective iterative method for solving large, sparse systems of linear equations numerically, provided that the coefficient matrix of which is symmetric and positive definite. This method is originally proposed by Hestenes and Stiefel [69]. In solving the linear equations $\mathbf{Ax} = \mathbf{b}$, the key idea of this method is to minimize the residual $\mathbf{r}_i = \mathbf{b} - \mathbf{Ax}_i$ along conjugate directions. This method solves the equation in at most n steps provided that there is no round-off error. A detailed explanation and development of this method could be found in [70, 71]. Later on, this method is extended to solve nonlinear systems of equations and unconstrained optimization problems such as potential energy minimization. In [72], Fletcher and Reeves developed an algorithm namely Fletcher-Reeves method by using the conjugate gradient method to minimize a general function. For a quadratic function, the method could locate the minimum in at most n steps apart from rounding off errors. For non-quadratic function, it usually takes more than n iterations to locate the minimum. Later, Polak and Ribière [73] modified the Fletcher-Reeves method by changing the way how the conjugate directions are calculated. Hestenes and Stiefel [69] also have their own approach to calculate the conjugate directions. For all the three methods, the conjugate search directions have a general form of

$$\mathbf{d}_k = \begin{cases} -\mathbf{g}_k & k = 1 \\ -\mathbf{g}_k + \beta_k \mathbf{d}_{k-1} & k \geq 2 \end{cases} \quad (2.8)$$

where \mathbf{g}_k is the gradient; β_k is a scalar. For the Fletcher-Reeves method, Polak-Ribière method, and Hestenes- Stiefel method, the β_k is defined by Eqns. (2.9), (2.10) and (2.11) respectively.

$$\beta_k^{FR} = \frac{\|\mathbf{g}_k\|^2}{\|\mathbf{g}_{k-1}\|^2} \quad (2.9)$$

$$\beta_k^{PR} = \frac{\mathbf{g}_k^T (\mathbf{g}_k - \mathbf{g}_{k-1})}{\|\mathbf{g}_{k-1}\|^2} \quad (2.10)$$

$$\beta_k^{HS} = \frac{\mathbf{g}_k^T (\mathbf{g}_k - \mathbf{g}_{k-1})}{\mathbf{d}_{k-1}^T (\mathbf{g}_k - \mathbf{g}_{k-1})} \quad (2.11)$$

Same as other iterative methods used in minimization problem, convergence is always the key issue. When the conjugate gradient is applied to a general function, a local quadratic approximation is always involved. In order to determine the step size in each conjugate direction for a general function, a line minimization search process is adopted, which could affect the convergence properties of the method depending on which convergence criteria is used for the line searches. When the method is used to solve unconstrained optimization problems, extensive research has been conducted on the convergence properties of this method. Zoutendijk [74] proved that the Fletcher-Reeves method with exact line searches is globally convergent on general functions. Al-Baali [75] extended Zoutendijk's conclusion to inexact line searches, particular for a variation of the Wolfe line-search. Powell [76] demonstrated that the Polak-Ribière and Hestenes-Stiefel methods are not globally convergent even with exact line searches. Meanwhile, Hu and Storey [77], and Gilbert and Nocedal [78] studied the global convergence of the algorithm related to the Fletcher-Reeves method with the strong Wolfe line search which indicates that the analysis is under the sufficient decent condition. In [78], Gilbert and Nocedal also studied the global convergence for the algorithms related to the Polak-Ribière method by considering different choice of β_k . Liu, Han and Yin [79] demonstrated the global convergence of the Fletcher-Reeves method under some conditions that are weaker than those in [75]. Dai and

Yuan proved that the Fletcher-Reeves method with the strong Wolfe line search is globally convergent provided that for each iteration, the search direction is downhill. Dai, Han, Liu., et al [80] demonstrated the global convergence of the Fletcher-Reeves- and Polak-Ribière-type methods without assuming the sufficient descent condition. This paper showed that the sufficient descent condition is no longer a need in the analysis of the global convergence of the Fletcher-Reeves and Polak-Ribière method.

2.4 Existing methods for degree reduction of Bézier curve

In this research, Bézier curve is used to represent the transition path. Bézier curve is a parametric curve which is defined by a set of control points. The number of control points determines the degree of the Bernstein polynomial basis functions that describe the shape of the curve. The curve interpolates its first and last control points and is tangent to the first and last sides of the open polygon defined by these control points [81]. We use degree elevation and reduction to maintain the space between control points during MEP search. Degree elevation of Bézier curve is exact. In contrast, degree reduction always has approximation involved. Degree reduction of Bézier curve is a process that uses a lower order curve to approximate a higher order curve. There are two major applications for the degree reduction of Bézier curve. One is to generate a piecewise linear approximation to a prescribed curve. The other is to transfer data from one geometric modeling system to another. During the process, usually a curve with higher degree must be approximated by several lower degree curves due to the limitation on the maximum polynomial degree that certain systems can store and work with. Intensive research has been conducted on the degree reduction of Bézier curve in order to minimize the error between the original and reduced curve. The approximation to the original curve depends heavily on the chosen distance or error function to be minimized. There are typically two categories of approaches. One is to approximate the shape for the curve. The other is to approximate polynomial function that defines the curve. It treats the degree reduction process as the inverse process of degree elevation. Hence, the degree reduction problem is shifted to the problem of solving an over-determined linear system for the polynomial coefficients that defines the curve. Watkins and Worsey [82] developed a degree reduction method which is based on minimax approximation techniques. The method looks at the degree reduction from the perspective of the curve itself and as a result can achieve a better approximation to the original curve measured in

the uniform norm. Eck [83] extended the degree reduction scheme originally proposed by Forrest [84] by introducing real weighting factors to blend the two set of coefficients together. It is a linear interpolation of the two set of coefficients. This shifts the problem of degree reduction to a problem of determining a set of real weighting factors. By minimizing the maximal Euclidean distance between the reduced curve and original curve based constrained Chebyshev polynomials, the weights could be determined. However, this algorithm requires intensive implementation effort because the constrained Chebyshev polynomials are known implicitly in general. In [85], Eck improved the algorithm by minimizing the least squares distance function using constrained Legendre polynomials. Bogacki and Weinstein [86] developed two algorithms (one-degree reduction and multiple degree reduction) that compute a constrained approximation of an n^{th} degree Bézier curve by an m^{th} ($m < n$) degree curve. The approximation is performed in the uniform norm applied component-wise with endpoint interpolation. Brunnett et al. [87] studied the optimal degree reduction (optimal approximation of an n^{th} degree Bézier curve by an m^{th} ($m < n$) degree curve) with respect to different norms, particularly to L_p norms and the uniform norm ($p = \infty$). Kim and Moon [88] addressed the degree reduction problem in the L^1 norm with endpoint interpolation. The scheme gives the best one-degree reduction of Bézier curve of the degree less than six with endpoint interpolation by using splines. For higher order curves, they proposed a scheme which is based on an appropriate transform of the Chebyshev polynomials of second kind. Kim and Ahn [89] developed a C^1 constrained degree reduction method using the constrained Jacobi polynomials, the coefficients of which are represented explicitly, as the error function for good degree reduction of Bézier curve. In [90], Ahn extended the C^1 constrained degree reduction method to C^k ($k = 2, 3$) constrained degree reduction using the constrained Jacobi polynomials.

In recent years, multi-degree reduction has been intensively studied. Instead of conducting the multi-degree reduction stepwise, methods are developed to perform multi-degree reduction at a time. Chen and Wang [91] developed a method named MDR by L_2 norm, which gives an explicit form of the least squares solution of multi-degree solution of Bézier curve with constraints of endpoints continuity. Sunwoo [92] generalized Chen and Wang's work [91] by finding an explicit form of the multi-degree reduction matrix for a Bézier curve with constraints

of endpoints continuity. The control points of the degree reduced curve can be expressed as a product of degree reduction matrix and the vector of original control points. Lu and Wang [93, 94] developed a multi-degree reduction method with G^2 continuity under L_2 norm. Later on, they developed another multi-degree reduction with respect to the $\sqrt{t-t^2}$ -weighted square norm by using the transformation matrix between Bernstein and Chebyshev basis. Rababah and Lee [95] developed a simple matrix form for r times degree reduction with respect to the weighted L_2 -norm by using the matrices of transformations between Chebyshev and Bernstein basis. Woźny and Lewanowicz [96] proposed a multi-degree reduction method under L_2 norm by using dual Bernstein basis polynomials. A re-parameterization-based multi-degree method is developed by Chen and Ma [97] recently, which introduces a piecewise linear function to replace the general t in the least square distance.

In this thesis, the developed algorithm uses the conjugate gradient method in searching the local minimums and the degree elevation and reduction in redistribution of the intermediate control points. The followed chapter presents the developed algorithm with a detailed explanation on how the conjugate gradient method and the degree reduction are used in the algorithm.

CHAPTER 3

CONCURRENT MULTI TRANSITION PATHWAY

SEARCH ALGORITHM

This chapter gives a detailed description of the proposed new algorithm. The objective of the algorithm developed here is to locate both the minimum energy positions and MEP simultaneously. So the precise positions of the stable configurations do not need to be known in advance. In addition, the algorithm is able to search multiple transition paths on the PES simultaneously, which gives us a comprehensive view of the energy landscape on the PES.

This global search algorithm includes three stages. The first stage involves the optimization of a single transition path, during which two local minima and one transition path that is close to the MEP will be located. The second stage is searching multiple transition paths starting from one single transition path obtained from the first stage. One curve is divided into two curves representing two stages of transition, which will be optimized in the same way as in the first stage. This stage will output several transition paths that approximate the true MEPs with those end points located at local minima. Then at the third stage, we let the control point with the maximum energy within each of those transition paths climb up in order to locate the actual saddle points. Sections 3.1 and 3.2 give a detailed description about the first two stages respectively. The procedure for this algorithm is summarized in Figure 1. An initial guess of the transition path is first provided. Then the path is optimized by minimizing the two end control points using the conjugate gradient method and meanwhile moving all the intermediate control points along the corresponding conjugate gradient directions. When two local minima are located by the end control points of the curve, the algorithm determines whether the curve crosses an extra local minimum or not. If not, the maximum energy point on the curve climbs up along the conjugate gradient directions to locate the true saddle point. If yes, the curve breaks into two new curves which represent initial guess for two new transition paths. Those two new curves are optimized and then checked following the same procedure as the initial curve. The check-and-break procedure continues until all the curves are unbreakable with their end control points locating at local minimums and one intermediate control point locating at the saddle point.

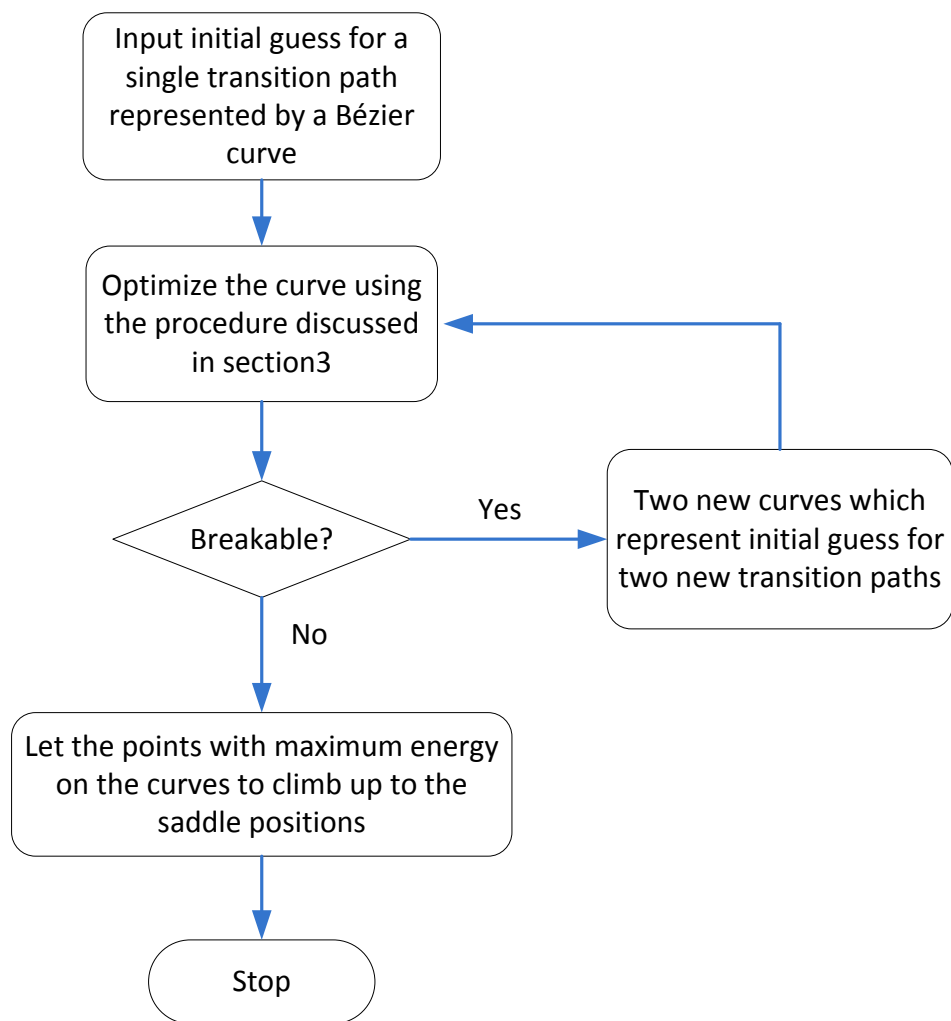


Figure 1: Flow chart for the procedure of the algorithm

3.1 A single transition pathway search

For the initial guess of a transition path which is represented by a single Bézier curve, the searching process for the stable configurations and the MEP is carried out in a sequential manner within a given iteration. A total of five control points are used for the initial curve. The more control points the curve has, the more accurate the search results will be, but with higher computational costs. The general process for a single transition pathway search is as follows. First, the two end control points of the curve are minimized by using the conjugate gradient method. Then, a set of conjugate directions for each intermediate control point is determined based on the new positions of the two end control points. Several minimization steps are applied

to each intermediate control points along their associated conjugate directions. After several iterations, the two end control points of the curve will gradually converge to the minimum energy positions and the curve will approach to the MEP. Figure 2 illustrates the procedure of searching a single transition path. Table 1 lists the pseudo-code of the algorithm for a single transition pathway search. The details are described in the following subsections.

3.1.1 Searching the Stable Configuration

As shown in Table 1, the local minimums are located by minimizing the two end control points of the curve iteratively. By definition the minimum energy location \mathbf{x}^* on the PES satisfies

$$\nabla V(\mathbf{x}^*) = 0 \quad (3.1)$$

where $V(\mathbf{x}^*)$ is the potential energy function with respect to the position vector \mathbf{x}^* in an n -dimensional configuration space; $\nabla V(\mathbf{x}^*)$ is the gradient of the potential on the PES at the location \mathbf{x}^* . The iterative location update during the minimization is given by

$$\mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} + \alpha^{(i)} \mathbf{d}^{(i)} \quad (3.2)$$

where $\alpha^{(i)}$ is the step length, and $\mathbf{d}^{(i)}$ is the search direction. The minimization process for the end points is carried out using the conjugate gradient method [72]. Detailed description about this method can be found in section 2.3. In our algorithm, the Fletcher-Reeves method is employed. The conjugate searching direction $\mathbf{d}^{(i)}$ in the i^{th} iteration are defined as a linear combination of $-\mathbf{g}^{(i)}$ and $\mathbf{d}^{(i-1)}$ [72],

$$\mathbf{d}^{(i)} = \begin{cases} -\mathbf{g}^{(i)} & \text{for } i = 1 \\ -\mathbf{g}^{(i)} + \frac{\|\mathbf{g}^{(i)}\|^2}{\|\mathbf{g}^{(i-1)}\|^2} \mathbf{d}^{(i-1)} & \text{for } i \geq 2 \end{cases} \quad (3.3)$$

The step size $\alpha^{(i)}$ is determined by using inexact line search along the corresponding conjugate directions. Namely, along each conjugate direction, several mini-steps are applied to the end points in order to locate the minimums along that direction. The minimum position in one conjugate gradient direction is the starting point for the corresponding followed conjugate

searching direction. In Table 1, in an n -dimensional search space, the search direction $\mathbf{D}^{(i)}$ in i^{th} iteration can be represented as

$$\alpha^{(i)}\mathbf{D}^{(i)} = \alpha_1^{(i)}\mathbf{d}_1^{(i)} + \dots + \alpha_n^{(i)}\mathbf{d}_n^{(i)} \quad (3.4)$$

For a quadratic potential function with n -dimensional inputs, the local minimum can be determined in at most n steps. For a non-quadratic function, local quadratic approximation is involved during the minimization process. For a non-quadratic function with n -dimensional inputs, it requires more than n steps to locate a minimum. For those functions, the conjugate searching directions which are built based on the Eq. (3.3) will gradually lose conjugacy when searching process continues, which could lead to divergence. In our algorithm, we recalculate the conjugate directions from one iteration to another, namely after n steps of conjugate search in order to avoid the divergence.

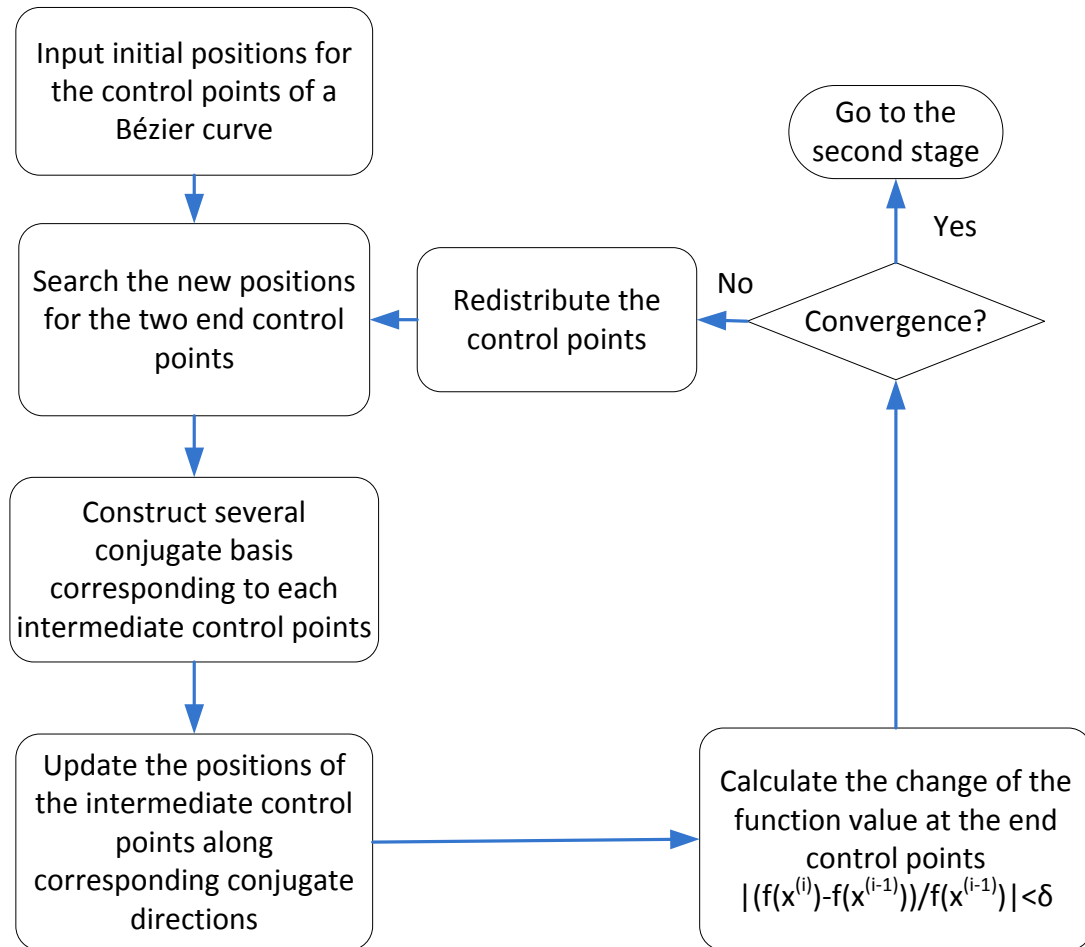


Figure 2: Flow chart for the procedure of a single transition path search

Table 1: Pseudo-code of the algorithm for a single transition pathway search

INPUT: Initial guess of a curve with control points of $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$.

OUTPUT: A curve with two end points located at two local minimums and the curve itself approaches to the MEP.

TOL= threshold for the percentage of change in potential energy value;

$V(\mathbf{p}_*^{(\#)})$ =potential energy value at points $\mathbf{p}_*^{(\#)}$;

$\mathbf{D}_0^{(i)}, \mathbf{D}_4^{(i)}$ = search direction at \mathbf{p}_0 and \mathbf{p}_4 respectively;

$\alpha_0^{(i)}, \alpha_4^{(i)}$ =step size for minimizing \mathbf{p}_0 and \mathbf{p}_4 respectively;

WHILE True

IF $\|(V(\mathbf{p}_0^{(i)}) - V(\mathbf{p}_0^{(i-1)})) / V(\mathbf{p}_0^{(i-1)})\| > TOL$ and $\|(V(\mathbf{p}_4^{(i)}) - V(\mathbf{p}_4^{(i-1)})) / V(\mathbf{p}_4^{(i-1)})\| > TOL$

$\mathbf{p}_0^{(i)} = \mathbf{p}_0^{(i-1)} + \alpha_0^{(i)} \mathbf{D}_0^{(i)}$; $\mathbf{p}_4^{(i)} = \mathbf{p}_4^{(i-1)} + \alpha_4^{(i)} \mathbf{D}_4^{(i)}$;

Minimize $\mathbf{p}_1^{(i-1)}, \mathbf{p}_2^{(i-1)}, \mathbf{p}_3^{(i-1)}$ in their associated conjugate directions to get a new set of intermediate control points $\mathbf{p}_1^{(i)}, \mathbf{p}_2^{(i)}, \mathbf{p}_3^{(i)}$ (see section 0).

IF There is zigzag along the curve

Do degree elevation or reduction locally (see Section 3.1.3 and 3.1.4).

END IF

ELSEIF $\|(V(\mathbf{p}_0^{(i)}) - V(\mathbf{p}_0^{(i-1)})) / V(\mathbf{p}_0^{(i-1)})\| < TOL$ and $\|(V(\mathbf{p}_4^{(i)}) - V(\mathbf{p}_4^{(i-1)})) / V(\mathbf{p}_4^{(i-1)})\| > TOL$

$\mathbf{p}_4^{(i)} = \mathbf{p}_4^{(i-1)} + \alpha_4^{(i)} \mathbf{D}_4^{(i)}$;

Minimize $\mathbf{p}_1^{(i-1)}, \mathbf{p}_2^{(i-1)}, \mathbf{p}_3^{(i-1)}$ in their corresponding conjugate directions to get a new set of intermediate control points $\mathbf{p}_1^{(i)}, \mathbf{p}_2^{(i)}, \mathbf{p}_3^{(i)}$.

IF There is zigzag along the curve

Do degree elevation or reduction locally (see Section 3.1.3 and 3.1.4).

END IF

ELSEIF $\|(V(\mathbf{p}_0^{(i)}) - V(\mathbf{p}_0^{(i-1)})) / V(\mathbf{p}_0^{(i-1)})\| < TOL$ and $\|(V(\mathbf{p}_4^{(i)}) - V(\mathbf{p}_4^{(i-1)})) / V(\mathbf{p}_4^{(i-1)})\| > TOL$

$\mathbf{p}_0^{(i)} = \mathbf{p}_0^{(i-1)} + \alpha_0^{(i)} \mathbf{D}_0^{(i)}$;

Minimize $\mathbf{p}_1^{(i-1)}, \mathbf{p}_2^{(i-1)}, \mathbf{p}_3^{(i-1)}$ in their corresponding conjugate directions to get a new set

of intermediate control points $\mathbf{p}_1^{(i)}, \mathbf{p}_2^{(i)}, \mathbf{p}_3^{(i)}$.

IF There is zigzag along the curve

Do degree elevation or reduction locally (see Section 3.1.3 and 3.1.4).

END IF

ELSE

Stop.

END IF

END WHILE

IF Two end points converge to the same local minimum

Re-input the initial guess of the control points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$.

END

3.1.2 Searching the MEP

Mathematically, for the n -dimension PES, the Hessian matrix \mathbf{H} (the matrix of the second derivative of the energy) at the first-order saddle points has one negative eigenvalue and $n-1$ positive ones. The eigenvectors \mathbf{s}_i form a conjugate basis (i.e. $\mathbf{s}_i^T \mathbf{H} \mathbf{s}_j = 0, \forall i \neq j$) with respect to the Hessian matrix. For a set of conjugate direction \mathbf{s}_i 's, in the vicinity of a first-order saddle point, there is one direction \mathbf{s}_0 along which the potential energy has a local maximum. For each of the other $n-1$ directions, the potential energy has a local minimum. The method presented here constructs a set of conjugate directions by making use of the Eq. (3.5) develop by Beale [56] which starts with a given arbitrary direction \mathbf{s}_0 . The rest conjugate directions are defined as

$$\begin{aligned} \mathbf{s}_1 &= -\mathbf{g}_1 + \frac{\mathbf{g}_1^T (\mathbf{g}_1 - \mathbf{g}_0)}{\mathbf{s}_0^T (\mathbf{g}_1 - \mathbf{g}_0)} \mathbf{s}_0 \\ \mathbf{s}_{i+1} &= -\mathbf{g}_{i+1} + \frac{\mathbf{g}_{i+1}^T (\mathbf{g}_1 - \mathbf{g}_0)}{\mathbf{s}_0^T (\mathbf{g}_1 - \mathbf{g}_0)} \mathbf{s}_0 + \frac{\mathbf{g}_{i+1}^T \mathbf{g}_{i+1}}{\mathbf{g}_i^T \mathbf{g}_i} \mathbf{s}_i, \quad i \geq 1 \end{aligned} \quad (3.5)$$

In this algorithm, for each intermediate control points, a set of corresponding conjugate directions are constructed by setting the \mathbf{s}_0 as the tangent direction approximated by the backward finite difference for the first half of the intermediate points and by the forward finite

difference for the second half respectively. For example, for the k^{th} control point \mathbf{p}_k ,

$$\mathbf{s}_0 = \begin{cases} \mathbf{p}_k - \mathbf{p}_{k-1} & \text{if } k \leq \left\lceil \frac{N}{2} \right\rceil \\ \mathbf{p}_k - \mathbf{p}_{k+1} & \text{if } k > \left\lceil \frac{N}{2} \right\rceil \end{cases} \quad (3.6)$$

where N is the total number of control points and $\lceil \cdot \rceil$ rounds up to an integer. In order to calculate \mathbf{s}_1 in Eq.(3.5), we first need to determine \mathbf{g}_0 and \mathbf{g}_1 . Here for the k^{th} control point \mathbf{p}_k , \mathbf{g}_0 is defined as the gradient at the middle point of the line segment connecting \mathbf{p}_k and its neighbor, namely,

$$\mathbf{g}_0 = \begin{cases} \nabla V\left(\frac{\mathbf{p}_{k-1} + \mathbf{p}_k}{2}\right) & \text{if } k < \left\lceil \frac{N}{2} \right\rceil \\ \nabla V\left(\frac{\mathbf{p}_k + \mathbf{p}_{k+1}}{2}\right) & \text{if } k > \left\lceil \frac{N}{2} \right\rceil \end{cases} \quad (3.7)$$

\mathbf{g}_1 is defined as the gradient at the position with maximum energy \mathbf{p}_{\max} along the direction \mathbf{s}_0 . Several steps of line maximization are applied to the point \mathbf{p}_k along \mathbf{s}_0 in order to locate \mathbf{p}_{\max} . Then several steps of line minimization along the conjugate direction \mathbf{s}_1 are applied to \mathbf{p}_k . The rest of the conjugate basis set are then built recursively using Eq. (3.5). Simultaneously, each time when a new conjugate direction is determined, several steps of line minimization along this direction are applied to the associated new positions of \mathbf{p}_{\max} .

3.1.3 Constrained Degree Elevation and Degree Reduction

After the evolution of the intermediate control points along the conjugate directions, those control points may become too close to each other. As a result, the control points only capture part of the information along the transition path. The resolution around the saddle region may be too low. This could lead to an underestimation of the energy barrier. Similar to the reparameterization process in the string method [49], a redistribution process of the control points after each evolution step is introduced in order to ensure that these intermediate control points are relatively well distributed. The degree elevation and reduction scheme for the Bézier curve

are employed to redistribute the intermediate control points in our algorithm.

Degree elevation increases the flexibility of a curve by introducing more degrees of freedom for control. By adding an extra control point to the definition of a Bézier curve, its degree is raised by one. The advantage of using the degree elevation technique is that we can increase the degree of a Bézier curve without changing its shape. The degree elevation of an n^{th} order Bézier curve by one produces an $(n+1)^{\text{th}}$ order Bézier curve with a new set of vertices \mathbf{q}_k defined by [98]

$$\begin{cases} \mathbf{q}_k = \mathbf{p}_k, & k = 0 \\ \mathbf{q}_k = \frac{k}{n+1}\mathbf{p}_{k-1} + \left(1 - \frac{k}{n+1}\right)\mathbf{p}_k, & k = 1, \dots, n \\ \mathbf{q}_k = \mathbf{p}_{k-1}, & k = n+1 \end{cases} \quad (3.8)$$

where \mathbf{p}_k 's are the original vertices of the n^{th} order Bézier curve. Eq. (3.8) can be written in a matrix form as

$$\mathbf{Q} = T_n \mathbf{P} \quad (3.9)$$

where $\mathbf{Q} = (\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_{n+1})^T$, $\mathbf{P} = (\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_n)^T$, and T_n is a $(n+2)$ by $(n+1)$ matrix defined as

$$T_n = \frac{1}{n+1} \begin{bmatrix} n+1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & n & 0 & \cdots & 0 & 0 & 0 \\ 0 & 2 & n-1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & n-1 & 2 & 0 \\ 0 & 0 & 0 & \cdots & 0 & n & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & n+1 \end{bmatrix}$$

The Bézier curve can be elevated more than one degree by applying Eq.(3.4) multiple times. In our algorithm, the curve is elevated only once within each iteration in order to make control points well distributed.

The purpose of degree elevation in our algorithm is to redistribute the intermediate control points. In other words, we are concerned more about how well the procedure makes the

control points distributed than about how small the error between the elevated curve and the original curve may have, as long as the introduced error is within a tolerance range. Based on those two considerations, a constraint is added to the original degree elevation scheme in order to better serve our purpose. When two control points become too close to each other after the degree elevation by Eq. (3.8), we manually set the new control point to be the arithmetic average of the two adjacent control points in the original curve. In other words, for each newly created control points of the elevated curve, we calculate the Euclidean distance between this point and the middle point of its corresponding adjacent points of the original curve. For the k^{th} control points \mathbf{q}_k of the elevated curve, if it satisfies the condition

$$\left\| \mathbf{q}_k - \frac{\mathbf{p}_{k-1} + \mathbf{p}_k}{2} \right\| > c \|\mathbf{p}_k - \mathbf{p}_{k-1}\| \quad (3.10)$$

where c ($0 < c < 1$) is a predefined constant, then \mathbf{q}_k is set as the middle point of the straight line $\overline{p_{k-1}p_k}$. Since it is too computationally expensive to keep elevating the curve recursively, degree reduction is introduced to keep a balance with degree elevation and maintain a reasonable computational cost.

Degree reduction approximates an n^{th} order Bézier curve with an m^{th} ($m < n$) order curve. Different from degree elevation, no exact degree reduction is possible in practice. So approximation is inevitable. Similar to some of the existing methods reviewed in section 2.4, we treat the degree reduction as an inverse process of the degree elevation. Equation (3.8) shows that the control points of an elevated Bézier curve can be exactly determined by the control points of the original Bézier curve through linear interpolation of the two adjacent points. For the degree reduction, we need to solve the over-determined system in Eq.(3.8) for the unknowns $\{\mathbf{p}_k\}_{k=0}^n$ as a linear combination of $\{\mathbf{q}_k\}_{k=0}^{n+1}$.

Many methods are available to solve Eq. (3.8) approximately. Here, we developed a reduction scheme similar to Eck's method [85] which solves the equations by three steps. In Eck's method, the Eq. (3.8) is solved first in the forward direction. That is, for $k = 1, \dots, n-1$, we receive

$$\mathbf{p}_k^I = \frac{1}{n+1-k} \left((n+1)\mathbf{q}_k - k\mathbf{p}_{k-1}^I \right) \quad (3.11)$$

where the superscript I indicates that the control points are obtained from the forward procedure. Then the control points $\{\mathbf{p}_k^I\}_{k=1}^{n-1}$ can be obtained recursively by setting $\mathbf{p}_{k-1}^I = \mathbf{q}_{k-1}$ when $k=1$, which indicates that the two end control points are fixed for the degree reduction. Then Eq.(3.8) is solved in the backward direction. That is, for $k=n, \dots, 2$, we receive

$$\mathbf{p}_{k-1}^{II} = \frac{1}{k} \left((n+1)\mathbf{q}_k - (n+1-k)\mathbf{p}_k^{II} \right) \quad (3.12)$$

where the superscript II indicates the control points is obtained from the backward procedure. And the control points $\{\mathbf{p}_k^{II}\}_{k=1}^{n-1}$ can be obtained recursively by setting $\mathbf{p}_k^{II} = \mathbf{q}_{k+1}$ when $k=n$. Thirdly, the unknown control points $\{\mathbf{p}_k\}_{k=0}^n$ of the reduced Bézier curve are calculated as a linear combination of the control points $\{\mathbf{p}_k^I\}_{k=1}^{n-1}$ and $\{\mathbf{p}_k^{II}\}_{k=1}^{n-1}$ as

$$\begin{cases} \mathbf{p}_k = \mathbf{q}_k & k = 0 \\ \mathbf{p}_k = (1-\lambda_k)\mathbf{p}_k^I + \lambda_k\mathbf{p}_k^{II} & k = 1, \dots, n-1 \\ \mathbf{p}_k = \mathbf{q}_k & k = n \end{cases} \quad (3.13)$$

where λ_k is the weights for \mathbf{p}_k^{II} .

The degree reduction problem is then converted to the one of determining the weights of the corresponding control points. In Eck's method, λ_k 's are determined by minimizing the least square distance between the original curve and the reduced curve, which is too costly for our purpose. Since the degree reduction in our algorithm is to redistribute the control points instead of transforming geometric information of curves which requires the error between the reduced curve and the original curve should be as small as possible. In order to reduce the computational cost, here the weights λ_k are defined as

$$\lambda_k = \frac{k}{n} \quad (k = 1, \dots, n-1) \quad (3.14)$$

The implementation test shows that this simplified degree reduction scheme makes the

distribution of the points worse for some cases. Sometimes it introduces loops, which is undesirable for our algorithm. Thus we developed a reduction scheme similar to Eck's [85] but with a modified forward and backward procedure. In order to determine the new control points for the reduced curve, we make use of the information of three adjacent points instead of one as in Eck's scheme from the original curve. The three-step procedure is described as follows. In the forward step, three sets of points are calculated by using

$$\mathbf{p}_{k,1}^I = \frac{1}{n+1-k} \left((n+1)\mathbf{q}_k - k\mathbf{p}_{k-1}^I \right) \quad (3.15)$$

$$\mathbf{p}_{k,2}^I = \frac{1}{n+1-k} \left((n+1)\mathbf{q}_{k+1} - k\mathbf{q}_k \right) \quad (3.16)$$

and

$$\mathbf{p}_{k,3}^I = \frac{1}{n+1-k} \left((n+1)\mathbf{q}_{k+2} - k\mathbf{q}_{k+1} \right) \quad (3.17)$$

where $k = 1, \dots, n-1$. Then an average of them

$$\mathbf{p}_k^I = \frac{\mathbf{p}_{k,1}^I + \mathbf{p}_{k,2}^I + \mathbf{p}_{k,3}^I}{3} \quad (3.18)$$

forms a new set of points $\{\mathbf{p}_k^I\}_{k=1}^{n-1}$. Similarly, in the backward step, a new set of control points

$\{\mathbf{p}_k^{II}\}_{k=1}^{n-1}$ can be obtained by using Eqs. (3.19), (3.20), (3.21) and (3.22).

$$\mathbf{p}_{k-1,1}^{II} = \frac{1}{k} \left((n+1)\mathbf{q}_k - (n+1-k)\mathbf{p}_k^{II} \right) \quad (3.19)$$

$$\mathbf{p}_{k-1,2}^{II} = \frac{1}{k} \left((n+1)\mathbf{q}_{k-1} - (n+1-k)\mathbf{q}_k \right) \quad (3.20)$$

$$\mathbf{p}_{k-1,3}^{II} = \frac{1}{k} \left((n+1)\mathbf{q}_{k-2} - (n+1-k)\mathbf{q}_{k-1} \right) \quad (3.21)$$

$$\mathbf{p}_{k-1}^{II} = \frac{\mathbf{p}_{k-1,1}^{II} + \mathbf{p}_{k-1,2}^{II} + \mathbf{p}_{k-1,3}^{II}}{3} \quad (3.22)$$

Finally, the new control points $\{\mathbf{p}_k\}_{k=0}^n$ can be obtained by using Eq.(3.13) and (3.14).

3.1.4 Local Degree Elevation and Degree Reduction

The degree elevation and reduction of a Bézier curve changes the shape of the curve globally, which will gradually smooth out the curve. Consequently, this prevents the curve from converging to a curved MEP. The remedy for this issue is to introduce a local degree elevation and reduction scheme. Within each iteration, we first check whether there is zigzag along the curve or not. If there is no zigzag along the curve, we do not do degree elevation and reduction to the curve. Otherwise, we do degree elevation and reduction locally based on the distribution of the zigzag. For example, for the k^{th} control point \mathbf{p}_k ($i = 1, \dots, n-1$), if it satisfies the condition

$$\arccos\left(\frac{\overline{P_{k-1}P_k} \cdot \overline{P_kP_{k+1}}}{\|P_{k-1}P_k\| \|P_kP_{k+1}\|}\right) > \beta \quad (3.23)$$

where β ($0 < \beta < \pi$) is a predefined constant, then it indicates that there is zigzag at the control point \mathbf{p}_k . We check each of the intermediate control points within each iteration. If there is no zigzag along the curve, degree elevation or reduction is not needed; otherwise, degree elevation or reduction is done locally. If the zigzag only exists within the first half of control points, degree elevation or reduction is only performed to the first half of control points. Similarly, it is performed only to the second half of control points if the zigzag only exists within the second half. If the zigzag exists in both, we do degree elevation or reduction globally.

3.2 Multiple Transition Pathway Search

Here, we present how to search multiple transition paths on the PES. Our algorithm starts with the initial guess of a single transition path. Once the local minimums are found as described in Section 3.1.1, this single path will be divided into two curves if an extra basin is located along the path. Both subdivided curves will then be treated individually and the algorithm will be applied to them. This subdivision process continues recursively until there is only one possible saddle point between any pair of local minimums. As a result, multiple local minimums and transition paths can be found within a target search area. Therefore, the initial guess of this single path should be set up such that the search area of interest can be covered.

During the multiple transition path search stage, a curve with two end control points located at the two local minimums obtained from the single transition path search will be

examined by using the curve subdivision scheme. It determines whether the curve crosses an extra basin with another local minimum. If yes, the curve is divided into two new curves at the intermediate control point that is located in the extra basin. Since the number of control points for those two newly created curves may be less than five, the degree elevation is applied to the two curves recursively until the number of control points for each curve reaches five. Those two elevated curves now represent the initial guesses for the two new transition paths. The elevated curves are optimized using the procedure listed in Table 1. After their respective local minimums are identified, the curve subdivision scheme is applied to them again. The check-and-break procedure continues until all of the curves are unbreakable with their end control points located at local minimums. By now, those curves are still the approximations of the individual MEPs. In order to find the actual energy barrier for each curve, the algorithm selects the control point with the maximum energy and makes it climb up to locate the saddle point. During the climbing process, a set of conjugate directions corresponding to the identified control point with the maximum energy are constructed. Different from the procedure in the single transition path search, the point with the maximum energy will be first maximized along s_0 direction, and then minimized along other directions s_i 's ($i \geq 1$). The same procedure in the single transition path search, i.e. minimization along directions with positive eigenvalues, is applied to the rest of intermediate control points during the climbing process. This further makes the curve converge to the MEP. Table 2 lists the pseudo-code of the algorithm for multiple transition path search. A curve with two end control points locating at two local minima is obtained from the single transition pathway search. The curve subdivision scheme is used to determine whether the curve crosses an extra local minimum or not. If not, energy point with maximum energy value on the curve climbs up along the conjugate gradient directions to locate the saddle point. If yes, the curve breaks into two new curves each of which has a total number of control points less than five. The degree elevation is applied to the two curves recursively until the number of control points for each curve reaches five. Those two elevated curves represent initial guess for two new transition paths. The elevated curves are optimized and then checked following the same procedure as the initial curve. The check-and-break procedure continues until all the curves are unbreakable with their end control points locating at local minima and one intermediate control point locating at the saddle point.

Table 2: Pseudo-code of the algorithm for multiple transition pathway search

INPUT: A curve $\varphi(\mathbf{x})$ with two end control points located at two local minimums.

OUTPUT: Multiple curves with their end points connected together locating at multiple local minimums. Besides, each curve has one point locating at the saddle point position.

N_i = number of newly produced curve for i^{th} iteration (N_0 is set to be 1).

$i = 0$

WHILE There exists newly produced curves in i^{th} iteration

$i = i + 1;$

$N_i = 0;$

FOR $j = 1, 2, \dots, N_{i-1}$

IF $\varphi_j(\mathbf{x})$ is breakable (using the scheme listed in Table 5)

Break the curve $\varphi_j(\mathbf{x})$ into two curves $\varphi_j^1(\mathbf{x})$ and $\varphi_j^2(\mathbf{x})$.

$N_i = N_i + 2;$

END IF

IF the number of control points for $\varphi_j^1(\mathbf{x})$ or $\varphi_j^2(\mathbf{x})$ is less than five

Do degree elevation to the curve $\varphi_j^1(\mathbf{x})$ or $\varphi_j^2(\mathbf{x})$.

END IF

Optimize $\varphi_j^1(\mathbf{x})$ and $\varphi_j^2(\mathbf{x})$ after degree elevation to get two optimized curve $\varphi_k(\mathbf{x})$ and $\varphi_{k+1}(\mathbf{x})$ ($k = N_i - 1$).

END FOR

END WHILE

FOR $j = 1, 2, \dots$, (total number of non-breakable curves produced during the WHILE loop)

Select the maxi-energy control point of the $\varphi_j(\mathbf{x})$ to climb up in order to locate the saddle point.

END FOR

The major step during the multiple transition path search is to determine some criteria of whether a curve is breakable and which intermediate control point we should select to break the

curve. Here the subdivision scheme for the fourth-order (with five control points) and fifth-order (with six control points) curves are used to demonstrate. If we use a curve with a degree lower than four, the limited number of control points may miss the detailed curvature information of the actual path on the PES. As a result, some of the local minimums will be missed. The subdivision scheme can be similarly extended to higher-order curves.

3.2.1 Scheme for Selecting Breakpoint

In this section, we present a curve subdivision scheme to determine whether a curve can be divided into two curves and which control point to be selected as the breakpoint for this breakable curve. This curve subdivision scheme is based on an assumption that the control points of a B ézier curve are relatively evenly distributed in a sequential manner. In other words, the curve itself has no loop or big curvature. We make use of the information of the gradient and potential energy value at each of the intermediate control points as well as their relative positions. Figure 3 shows a B ézier polygon on the PES with two end control points located at the minima of two separate basins of local minima. $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ and \mathbf{p}_4 are control points. $-\nabla V(\mathbf{p}_1)$,

$-\nabla V(\mathbf{p}_2)$, and $-\nabla V(\mathbf{p}_3)$ illustrate the negative gradient directions at the position $\mathbf{p}_1, \mathbf{p}_2$, and \mathbf{p}_3 respectively. θ_1, θ_2 , and θ_3 are the angles between the negative gradient and the control polygon.

By examining the three angles as well as the potential energy values at those intermediate control points, it is able to determine whether the curve crosses a third basin of local minimums. There are a total of eight combinations with the angle distributions. The process of this scheme includes three steps. The first step is to check the combination of θ_1 and θ_3 . If no conclusion can be reached, a second step is to check θ_2 . If we still cannot decide by the second step, the energy values at the intermediate points will be considered as the third step.

Table 3 summarizes the curve subdivision scheme for a fourth order curve. The details about the scheme to determine a breakpoint for a fourth-order curve is described in the remainder of this section.

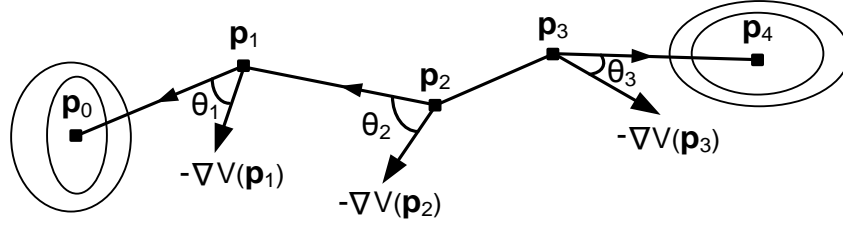


Figure 3: Illustration for multiple pathway search (five control points)

The first step of the process is to check the angles θ_1 and θ_3 . If both θ_1 and θ_3 are larger than $\pi/2$ (i.e. Case 2 and Case 3 in Table 3), not only \mathbf{p}_0 and \mathbf{p}_1 are in different basins of local minimums but also \mathbf{p}_3 and \mathbf{p}_4 , which indicates that the curve crosses at least a third basin of local minimum. Any of the three intermediate control points could be a breakpoint. In our algorithm, we choose \mathbf{p}_2 as the breakpoint. If either θ_1 or θ_3 is less than $\pi/2$ (i.e. Case 1, 4, 5, 6, 7, and 8 in Table 3), it is not guaranteed that the curve would go through a third basin of local minimum by checking θ_1 and θ_3 only. For example, when θ_1 is larger than $\pi/2$ and θ_3 is less than $\pi/2$, there are two sets of possible positions for the control points, i.e. Case 1 and 4. Since θ_3 is less than $\pi/2$, \mathbf{p}_3 and \mathbf{p}_4 could be located in the same basin. If \mathbf{p}_2 is located in a different basin from \mathbf{p}_3 , the curve crosses the third basin. Otherwise, \mathbf{p}_1 , \mathbf{p}_2 , \mathbf{p}_3 , and \mathbf{p}_4 could be in the same basin and the curve crosses only two adjacent basins. Therefore, we are unable to decide whether the curve is breakable or not with the only information that θ_1 is larger than $\pi/2$ and θ_3 is less than $\pi/2$. More information is required.

As a second step, we take \mathbf{p}_2 into consideration by checking θ_2 . Here we use Cases 1 and 4 to illustrate. When θ_2 is less than $\pi/2$ (Case 1 in Table 3), it indicates that \mathbf{p}_2 cannot be located in the same basin as \mathbf{p}_3 and \mathbf{p}_4 . Also as discussed in the first step, \mathbf{p}_0 and \mathbf{p}_1 are located in two different basins as in Case 1. Thus the curve should cross at least a third basin. Either \mathbf{p}_1 or \mathbf{p}_2 could be a breakpoint. Here, we select \mathbf{p}_2 to break. When θ_2 is larger than $\pi/2$ (Case 4 in

Table 3), the negative gradients at the position \mathbf{p}_1 , \mathbf{p}_2 , and \mathbf{p}_3 are in the similar directions. \mathbf{p}_1 , \mathbf{p}_2 , \mathbf{p}_3 , and \mathbf{p}_4 could be in the same basin which means that the curve crosses only two adjacent basins of local minima. Thus we need further information to determine if the curve is breakable.

In the third step, the potential energy values at \mathbf{p}_1 , \mathbf{p}_2 , and \mathbf{p}_3 are considered. If the potential energy values at positions \mathbf{p}_1 , \mathbf{p}_2 , and \mathbf{p}_3 have the monotonic relationship $V(\mathbf{p}_1) > V(\mathbf{p}_2) > V(\mathbf{p}_3)$, \mathbf{p}_1 , \mathbf{p}_2 , and \mathbf{p}_3 are considered as in the same basin, although there is still a slight chance that they are not. The curve is defined as unbreakable under this condition; otherwise, we break up the curve at the point \mathbf{p}_1 .

The above three-step procedure for Cases 1 and 4 can be extended to Cases 5, 6, 7, and 8. For Cases 5 and 8, θ_1 is less than $\pi/2$ and θ_3 is larger than $\pi/2$. When θ_2 is larger than $\pi/2$ (Case 8 in Table 3), the curve is breakable at the points \mathbf{p}_2 and \mathbf{p}_3 . Here we select \mathbf{p}_2 as the break point. When θ_2 is less than $\pi/2$ (Case 8 in Table 3), and $V(\mathbf{p}_3) > V(\mathbf{p}_2) > V(\mathbf{p}_1)$, the curve is unbreakable; otherwise, we break it at \mathbf{p}_3 . When both θ_1 and θ_3 are less than $\pi/2$ (Cases 6 and 7 in Table 3), the additional information of θ_2 does not help to determine. Hence we use the potential energy value directly. When a curve crosses two adjacent basins and the control points are relatively evenly distributed, the energy level at the middle point should be the largest. Based on this fact, when $V(\mathbf{p}_2) > V(\mathbf{p}_1)$ and $V(\mathbf{p}_2) > V(\mathbf{p}_3)$, the curve is defined as unbreakable in the algorithm; otherwise, at the break point is chosen as \mathbf{p}_2 .

The above procedure for breaking a fourth order curve can be extended to higher order curves. As an example, Table 4 summarizes the curve subdivision scheme for a fifth order curve. The discussion of the algorithm in this thesis is based on a fourth order curve. As an example, Table 5 lists the pseudo-code for breaking a curve with five and six control points.

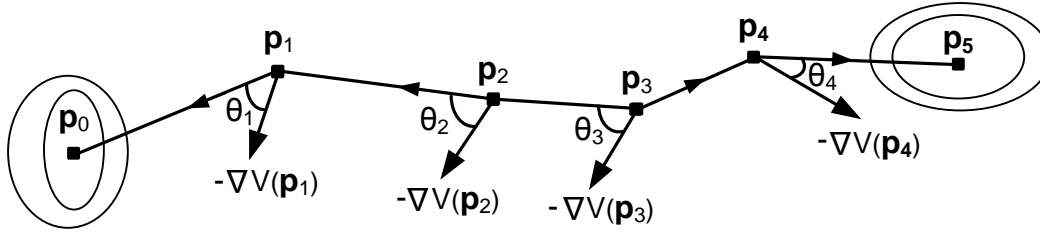


Figure 4: Illustration for multiple pathway search (six control points)

Table 3: Curve subdivision scheme (five control points)

Eight Cases		Greater (>) or smaller (<) than $\pi/2$			Breakable ?
		θ_1	θ_2	θ_3	
1		>	<	<	Break at \mathbf{p}_2
2		>	<	>	Break at \mathbf{p}_2
3		>	>	>	Break at \mathbf{p}_2
4		>	>	<	If $V(\mathbf{p}_1) > V(\mathbf{p}_2) > V(\mathbf{p}_3)$, the curve is defined as unbreakable; Otherwise, break at \mathbf{p}_1 .
5		<	<	>	If $V(\mathbf{p}_3) > V(\mathbf{p}_2) > V(\mathbf{p}_1)$, the curve is defined as unbreakable; Otherwise, break at \mathbf{p}_3
6		<	<	<	If $V(\mathbf{p}_1) < V(\mathbf{p}_2)$ and $V(\mathbf{p}_3) < V(\mathbf{p}_2)$, the curve is defined as unbreakable; Otherwise, break at \mathbf{p}_2
7		<	>	<	Same as case 6
8		<	>	>	Break at \mathbf{p}_2

Table 4: Curve subdivision scheme (six control points)

Sixteen Cases		Greater (>) or smaller (<) than $\pi/2$				Breakable?
		θ_1	θ_2	θ_3	θ_4	
1		>	<	<	>	Break at \mathbf{p}_2
2		>	<	>	>	Break at \mathbf{p}_2
3		>	>	>	>	Break at \mathbf{p}_2
4		>	>	<	>	Break at \mathbf{p}_2
5		>	<	<	<	Break at \mathbf{p}_2
6		>	<	>	<	Break at \mathbf{p}_3
7		>	>	>	<	Break at \mathbf{p}_3
8		>	>	<	<	If $V(\mathbf{p}_1) > V(\mathbf{p}_2) > V(\mathbf{p}_3) > V(\mathbf{p}_4)$, the curve is defined as unbreakable; otherwise, break at \mathbf{p}_2
9		<	>	<	>	Break at \mathbf{p}_3
10		<	>	>	>	Break at \mathbf{p}_3
11		<	<	<	>	Break at \mathbf{p}_3
12		<	<	>	>	If $V(\mathbf{p}_4) > V(\mathbf{p}_3) > V(\mathbf{p}_2) > V(\mathbf{p}_1)$, the curve is defined as unbreakable; otherwise, break at \mathbf{p}_4
13		<	<	<	<	If $V(\mathbf{p}_3) > V(\mathbf{p}_4)$ and $V(\mathbf{p}_2) > V(\mathbf{p}_1)$, the curve is defined as unbreakable; otherwise, if $V(\mathbf{p}_2) < V(\mathbf{p}_1)$, break at \mathbf{p}_2 , else break at \mathbf{p}_3
14		<	<	>	<	If $V(\mathbf{p}_3) > V(\mathbf{p}_2) > V(\mathbf{p}_1)$, the curve is defined as unbreakable; otherwise, break at \mathbf{p}_3
15		<	>	>	<	Break at \mathbf{p}_2
16		<	>	<	<	If $V(\mathbf{p}_2) > V(\mathbf{p}_3) > V(\mathbf{p}_4)$, the curve is defined as unbreakable; otherwise, break at \mathbf{p}_2

Table 5: Pseudo-code of the curve division scheme (five and six control points)

INPUT: An optimized curve $\varphi(\mathbf{x})$ with two end control points located at two local minimums.

OUTPUT: Two curve sections of $\varphi(\mathbf{x})$

IF $\varphi(\mathbf{x})$ has five control points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$ (refer to Figure 3)

IF $\theta_1 > \pi/2$ and $\theta_3 > \pi/2$

\mathbf{p}_2 is selected as breakpoint

ELSEIF $\theta_1 > \pi/2$ and $\theta_3 < \pi/2$

IF $\theta_2 < \pi/2$

\mathbf{p}_2 is selected as breakpoint

ELSE

IF $V(\mathbf{p}_1) > V(\mathbf{p}_2) > V(\mathbf{p}_3)$

$\varphi(\mathbf{x})$ is non-breakable

ELSE

\mathbf{p}_1 is selected as breakpoint

END IF

END IF

ELSEIF $\theta_1 < \pi/2$ and $\theta_3 > \pi/2$

IF $\theta_2 > \pi/2$

\mathbf{p}_2 is selected as breakpoint

ELSE

IF $V(\mathbf{p}_3) > V(\mathbf{p}_2) > V(\mathbf{p}_1)$

$\varphi(\mathbf{x})$ is non-breakable

ELSE

\mathbf{p}_1 is selected as breakpoint

END IF

END IF

ELSEIF $\theta_1 < \pi/2$ and $\theta_3 < \pi/2$

IF $V(\mathbf{p}_2) > V(\mathbf{p}_1)$ and $V(\mathbf{p}_2) > V(\mathbf{p}_3)$

$\varphi(\mathbf{x})$ is non-breakable

ELSE

\mathbf{p}_2 is selected as breakpoint

END IF

END IF

ELSEIF $\varphi(\mathbf{x})$ has six control points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5$ (refer to Figure 4)

IF $\theta_1 > \pi/2$ and $\theta_4 > \pi/2$

\mathbf{p}_2 is selected as breakpoint

ELSEIF $\theta_1 > \pi/2$ or $\theta_4 < \pi/2$

IF Either $\theta_2 < \pi/2$ or $\theta_3 < \pi/2$

\mathbf{p}_2 is selected as breakpoint

ELSE

IF $V(\mathbf{p}_1) > V(\mathbf{p}_2) > V(\mathbf{p}_3) > V(\mathbf{p}_4)$

$\varphi(\mathbf{x})$ is non-breakable

ELSE

\mathbf{p}_1 is selected as breakpoint

END IF

END IF

ELSEIF $\theta_1 < \pi/2$ and $\theta_4 > \pi/2$

IF Either $\theta_2 > \pi/2$ or $\theta_3 > \pi/2$

\mathbf{p}_3 is selected as breakpoint

ELSE

IF $V(\mathbf{p}_4) > V(\mathbf{p}_3) > V(\mathbf{p}_2) > V(\mathbf{p}_1)$

$\varphi(\mathbf{x})$ is non-breakable

ELSE


```

    p4 is selected as breakpoint
END IF
END IF
ELSEIF  $\theta_1 < \pi/2$  and  $\theta_4 < \pi/2$ 
    IF  $\theta_2 < \pi/2$  and  $\theta_3 > \pi/2$ 
        IF  $V(\mathbf{p}_2) > V(\mathbf{p}_1)$  and  $V(\mathbf{p}_3) > V(\mathbf{p}_4)$ 
             $\varphi(\mathbf{x})$  is non-breakable
        ELSE
            IF  $V(\mathbf{p}_1) > V(\mathbf{p}_2)$ 
                p2 is selected as breakpoint
            ELSEIF  $V(\mathbf{p}_4) > V(\mathbf{p}_3)$ 
                p3 is selected as breakpoint
            END IF
        END IF
    ELSEIF  $\theta_2 < \pi/2$  and  $\theta_3 < \pi/2$ 
        IF  $V(\mathbf{p}_3) > V(\mathbf{p}_2) > V(\mathbf{p}_1)$ 
             $\varphi(\mathbf{x})$  is non-breakable
        ELSE
            p3 is selected as breakpoint
        END IF
    ELSEIF  $\theta_2 > \pi/2$  and  $\theta_3 > \pi/2$ 
        IF  $V(\mathbf{p}_2) > V(\mathbf{p}_3) > V(\mathbf{p}_4)$ 
             $\varphi(\mathbf{x})$  is non-breakable
        ELSE
            p2 is selected as breakpoint
        END IF
    ELSEIF  $\theta_2 > \pi/2$  and  $\theta_3 < \pi/2$ 

```

```


$\mathbf{p}_2$  is selected as breakpoint



END IF



END IF



END IF


```

3.2.2 Discussion on the subdivision scheme

The proposed curve subdivision scheme for selecting the breakpoint is not perfect. It could treat some breakable curves as unbreakable ones. For example, for a curve with five control points, when $\theta_1 > \pi/2$, $\theta_3 < \pi/2$ and $V(\mathbf{p}_1) > V(\mathbf{p}_2) > V(\mathbf{p}_3)$, we define the curve as unbreakable. It is true if the curve only passes through two adjacent basins of local minima. But if the curve covers a long range with several extra local minima, there is still a small chance that the control points are positioned in the manner which satisfies the unbreakable conditions. The scheme will treat both of the two curves as unbreakable. A remedy for missing breakable curves is adding an extra step to double check each unbreakable curve for one more time. If a curve is identified as unbreakable curve for the first time, the control points of the curve will be redistributed by using degree elevation or degree reduction. Then this elevated or reduced curve will be checked again to see whether it is breakable. This extra step will increase the accuracy of subdivision but also with extra computational cost.

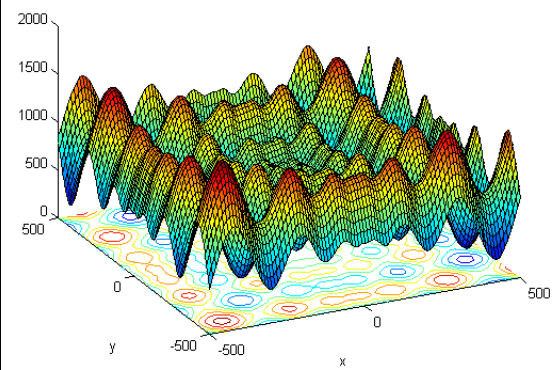
CHAPTER 4

IMPLEMENTATION AND DEMONSTRATION

This chapter demonstrates the proposed concurrent search algorithm for multiple phase transition pathways. First, we test the algorithm of a single transition pathway on LEPS potential and LEPS plus harmonic oscillator potential [5, 99]. These two potential functions are two-dimensional benchmark problems which are frequently used to test transition pathway search methods. These two potential model mimics a reaction involving three atoms. Details will present in Section 4.1 and 4.2. Then we demonstrate the proposed multiple transition pathway search algorithm by applying it to search the saddle points and local minimums on two different two-dimensional PESs defined by the Rastrigin function and Schwefel function respectively. The implementation was done using MATLAB. Source codes are included in the Appendix. Table 6 lists the definition and graphic in two-dimension for the four test functions. For LEPS potential, the Q functions illustrate Coulomb interactions between the electron clouds and the nuclei. The J functions illustrate the quantum mechanical exchange interactions. The parameters for the LEPS potential are defined as $a = 0.05$, $b = 0.30$, $c = 0.05$, $d_{AB} = 4.746$, $d_{BC} = 4.746$, $d_{AC} = 3.445$, $r_0 = 0.742$, and $\alpha = 1.942$. For LEPS plus harmonic oscillator potential, the parameters are defined as $r_{AC} = 3.742$, $k_c = 0.2025$, and $c = 1.154$. All the other parameters are the same as the ones defined for LEPS potential except $b = 0.80$ in LEPS plus harmonic oscillator potential.

Table 6: Test functions

Function	Definition	Graphic in two-dimension
LEPS potential	$V^{LEPS}(r_{AB}, r_{BC}) = \frac{Q_{AB}}{1+a} + \frac{Q_{BC}}{1+b} + \frac{Q_{AC}}{1+c} - \left[\frac{J_{AB}^2}{(1+a)^2} + \frac{J_{BC}^2}{(1+b)^2} + \frac{J_{AC}^2}{(1+c)^2} - \frac{J_{AB}J_{BC}}{(1+a)(1+b)} - \frac{J_{BC}J_{AC}}{(1+b)(1+c)} - \frac{J_{AB}J_{AC}}{(1+a)(1+c)} \right]^{1/2}$ <p>where</p> $Q(r) = \frac{d}{2} \left(\frac{3}{2} e^{-2\alpha(r-r_0)} - e^{-\alpha(r-r_0)} \right)$ $J(r) = \frac{d}{4} \left(e^{-2\alpha(r-r_0)} - 6e^{-\alpha(r-r_0)} \right)$	
LEPS + Harmonic oscillator potential	$V(r_{AB}, x) = VLEPS(r_{AB}, r_{AC} - r_{AB}) + 2k_c (r_{AB} - (r_{AC} / 2 - x / c))^2$	
Rastrigin	$f(\mathbf{x}) = 10n + \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i)]$	

Schwefel	$f(\mathbf{x}) = 418.9829n - \sum_{i=1}^n x_i \sin \sqrt{ x_i }$	
----------	--	--

4.1 Test result for LEPS potential

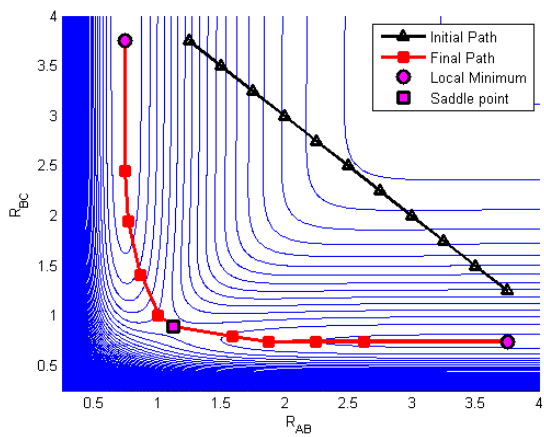
The LEPS potential model mimics a reaction involving three atoms A, B, and C constricted to motion along a straight line. There is only one bond formed, either between atoms A and B or between B and C [5, 99]. The detailed description and a 3-D graphic of the potential function can be found in Table 6. We test the single transition pathway search algorithm on this function by using two different initial positions of the transition path. The constant coefficient of the step size for minimizing the end control points and moving the intermediate control points is set to be 0.01 and 0.025 respectively. The results are illustrated in Figure 5 using contour plot. The black line represents the initial path. The red line represents the final path identified by using the algorithm listed in Table 1. The purple circle markers indicate the position of local minimums, while the purple square marker represents the position of saddle point.

For different initial positions (refer to (a) and (b) in Figure 5), the algorithm locates the same saddle point while it locates different local minimums. The result for locating different local minimums is sensitive to the initial positions. This is due to the characteristic of the LEPS potential function. The 3-D graphic for LEPS potential function in Table 6 shows that there is a long flat valley around the local minimum region. Each point along the valley could be a potential local minimum. The algorithm will stop searching local minimums as long as it locates one of the potential local minimums. Starting from different initial positions, the algorithm will follow different searching path. As a result, it will locate different potential local minimums. That explains why the located local minimums are different for different initial positions.

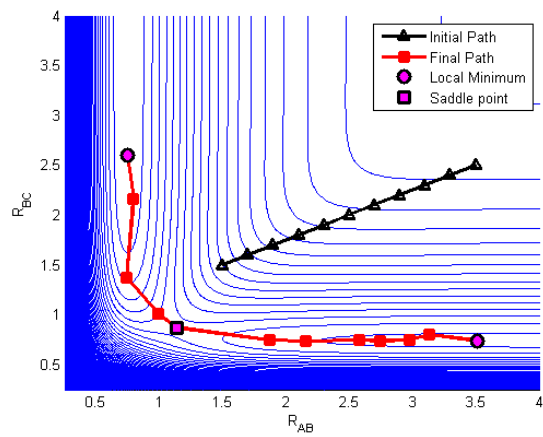
For the initial position in Figure 5 (c), the algorithm failed to locate the saddle point. The two end control points tend to converge to the same local minimum. This is because that the conjugate gradient method is a local search method. The control point will converge to the nearest local optimal point. Since in the algorithm developed in this thesis, the two end control points are optimized independently based on conjugate gradient method, the two end control points will converge to the same local minimums when the initial positions of the two points are close to the same local minimum.

4.2 Test result for LEPS plus harmonic oscillator potential

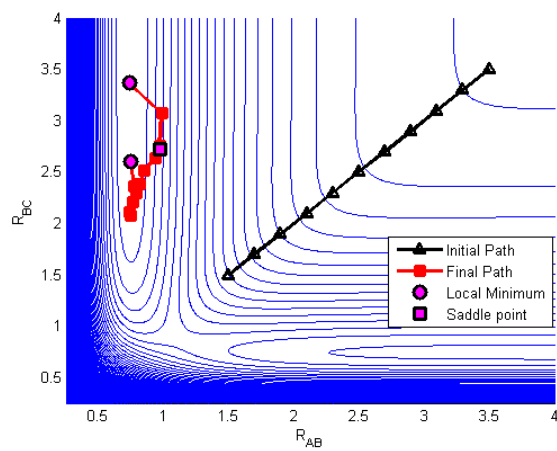
Different with the LEPS potential model, the location of the two end atoms A and C in this model is fixed. Only atom B is allowed to move. In addition, this model introduces an additional degree of freedom which can be interpreted as a fourth atom that is coupled in a harmonic way with the atom B [5, 99]. The detailed description and a 3-D graphic of the potential function can be found in Table 6. We test the single transition pathway search algorithm on this function by using three different initial positions of the transition path. The constant coefficient of the step size for minimizing the end control points and moving the intermediate control points is set to be 1/35 and 1/45 respectively. The results are illustrated in Figure 6 using contour plot. The black line with triangle markers represents the initial path. The red line with square markers represents the final path identified by using the algorithm listed in Table 1. The purple circle markers indicate the position of local minimums, while the purple square marker represents the position of saddle point. The results show that the algorithm is able to locate the local minima and saddle points for different initial positions.



(a)



(b)



(c)

Figure 5: Test results for LEPS potential function

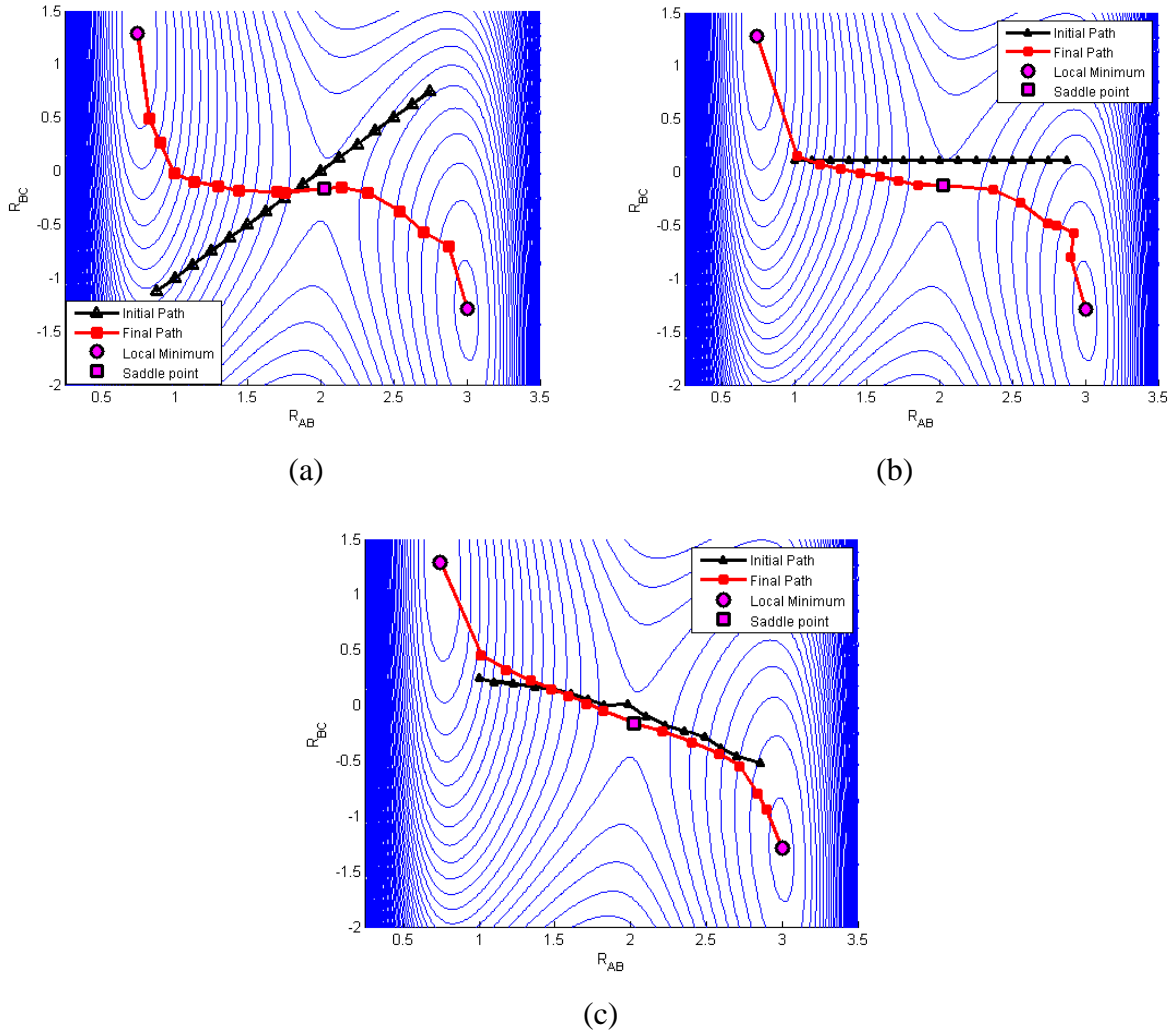


Figure 6: Test results for LEPS plus harmonic oscillator potential function

4.3 Test result for Rastrigin function

The Rastrigin function is a non-convex function frequently used to test the optimization algorithm. The function has a global minimum at $\mathbf{x}=(0,\dots,0)$ as well as several local minima. As discussed in Section 3.2, the initial guess for the transition path should be a curve with five control points which are relatively evenly distributed. Here, we choose a curve $\varphi(\mathbf{x})$ with five control points located at $(-2.81, 0.50)$, $(-1.43, 2.90)$, $(0.23, -2.47)$, $(1.57, 2.67)$, and $(2.91, -0.11)$, which are visualized in Figure 7 as ‘initial path’. First,

the optimization procedure listed in Table 1 is applied to $\varphi(\mathbf{x})$, which produces a curve $\varphi'(\mathbf{x})$ with two end control points located at the two local minimums. Then the multi-transition pathway search algorithm listed in Table 2 is applied to $\varphi'(\mathbf{x})$. A total of seven local minimums and six corresponding saddle points are located by this algorithm. The positions of those local minimums and the corresponding saddle points are listed in Table 7. Figure 7 shows the result using contour graphs. In order to test the robust of the algorithm, we test the algorithm by using a set of different initial positions located at $(-2.81, -1.50)$, $(-1.43, -1.50)$, $(0.23, -1.50)$, $(1.57, -1.50)$, and $(2.91, -1.50)$, which are shown in Figure 8 as black dots. The result shows that the algorithm performs well for different initial positions. For the second set of initial positions, a total of seven local minimums and six corresponding saddle points are located. The details about the local minimums and corresponding saddle points are listed in Table 8. The result is also illustrated in Figure 8.

Table 7: Test results on Rastrigin function (contour plot refer Figure 7)

Path No	Local minimums		Saddle
1	$(-2.9849, 0)$	$(-1.9899, 0)$	$(-2.5516, 0.0201)$
2	$(-1.9899, 0)$	$(-0.9950, 0)$	$(-1.5484, 0.0210)$
3	$(-0.9950, 0)$	$(0, 0)$	$(-0.5345, -0.0133)$
4	$(0, 0)$	$(0.9950, 0)$	$(0.4656, -0.0116)$
5	$(0.9950, 0)$	$(1.9899, 0)$	$(1.4688, -0.0132)$
6	$(1.9899, 0)$	$(2.9849, 0)$	$(2.4742, 0.0472)$

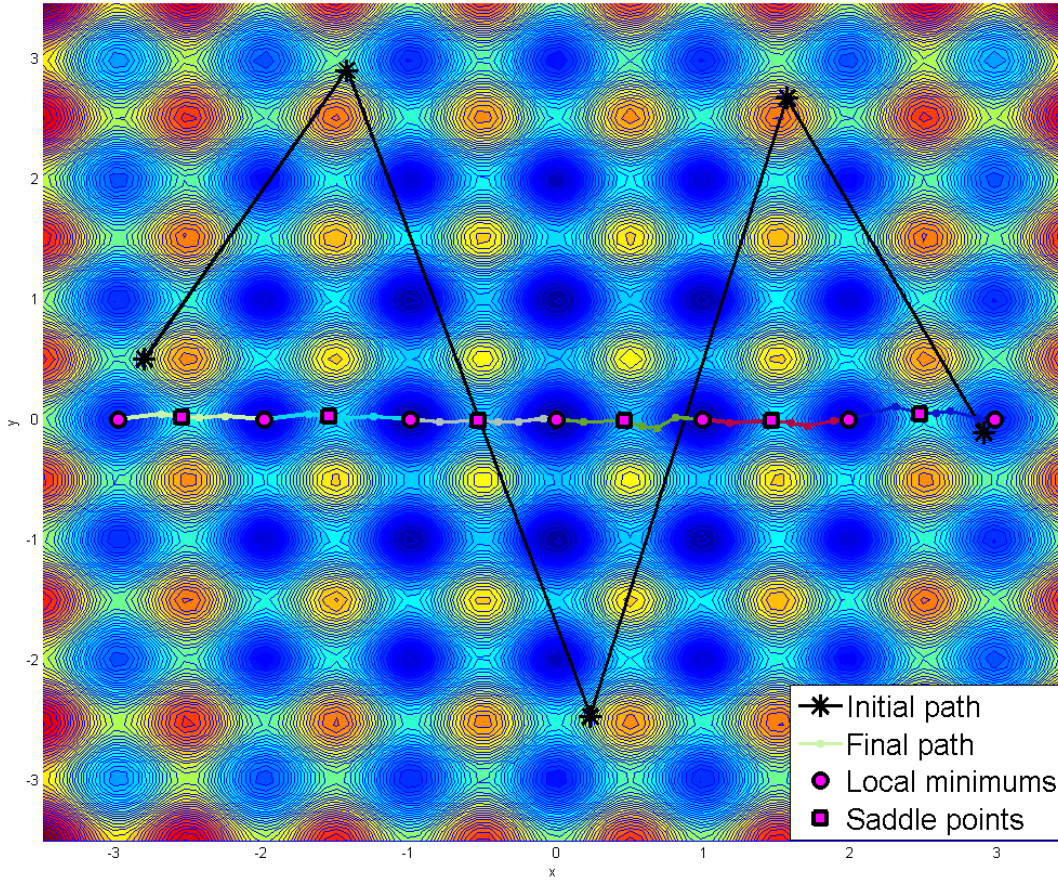


Figure 7: Test result for Rastrigin function with the initial position at $(-2.81, 0.50)$, $(-1.43, 2.90)$, $(0.23, -2.47)$, $(1.57, 2.67)$, and $(2.91, -0.11)$.

Table 8: Test results on Rastrigin function (contour plot refer Figure 8)

Path No	Local minimums		Saddle
1	$(-2.9849, -0.9950)$	$(-1.9899, -0.9950)$	$(-2.5497, -0.9832)$
2	$(0, -0.9950)$	$(0.9950, -0.9950)$	$(0.4591, -0.9764)$
3	$(-1.9899, -0.9950)$	$(-0.9950, -0.9950)$	$(-1.5496, -0.9776)$
4	$(-0.9950, -0.9950)$	$(0, -0.9950)$	$(-0.5441, -0.9738)$
5	$(0.9950, -0.9950)$	$(1.9899, -0.9950)$	$(1.5399, -0.9666)$
6	$(1.9899, -0.9950)$	$(2.9849, -0.9950)$	$(2.5491, -0.9816)$

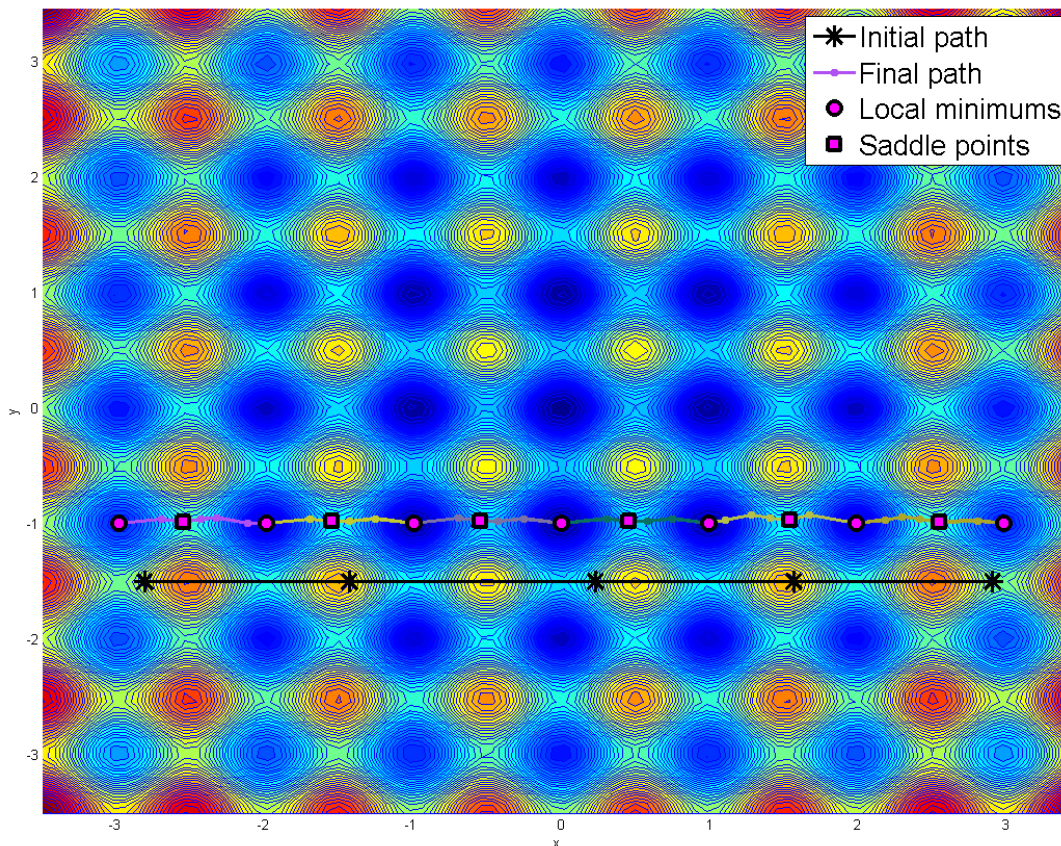


Figure 8: Test result for Rastrigin function with the initial position at $(-2.81, -1.50)$, $(-1.43, -1.50)$, $(0.23, -1.50)$, $(1.57, -1.50)$, and $(2.91, -1.50)$.

4.4 Test result for Schwefel function

Section 4.1 shows that the algorithm works very well on the PES defined by Rastrigin function. But the contour of the Rastrigin function is uniformly distributed as we can see from the contour plot in Figure 7. In real application, most of the potential energy surfaces are non-uniform. Thus the Schwefel function which has a relatively non-uniform potential energy surface is selected to test our algorithm. We test our algorithm on the Schwefel surface following the same procedure as we did on Rastrigin surface. We also test the algorithm with two set of initial positions. The first initial positions are located at $(-100.3, 25)$, $(-40.5, 40)$, $(17.8, -10)$, $(69.8, 70.6)$, and $(130.2, 98.7)$, which are illustrated in Figure 9 with black dots. A total of six

local minimums and five corresponding saddle points are located. Details are listed in Table 9. Also, the results are visualized in Figure 9. The second initial positions are located at $(-100.3, -70)$, $(-40.5, -70)$, $(17.8, -70)$, $(69.8, -70)$, and $(130.2, -70)$, as shown in Figure 10 with black dots. A total of five local minimums and four corresponding saddle points are located. Details are listed in Table 10. The results are visualized in Figure 10.

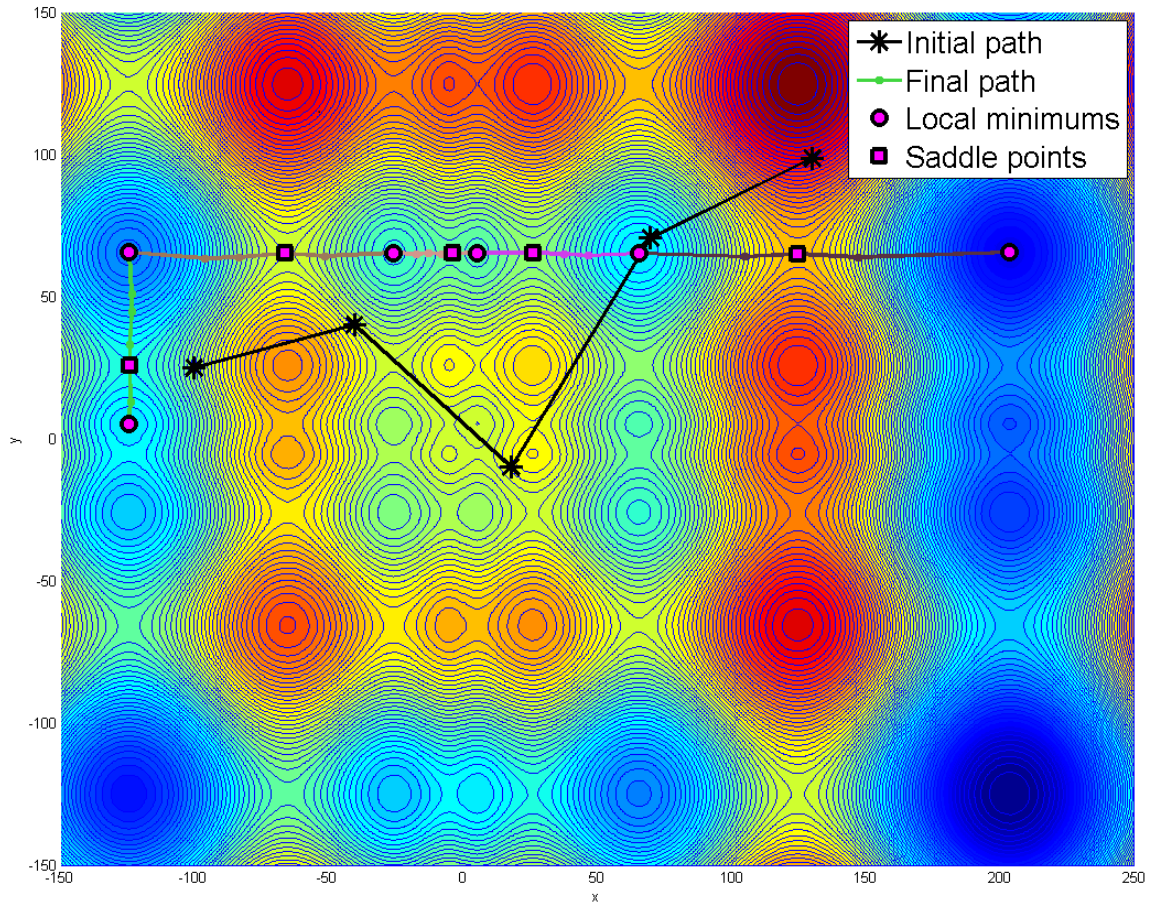


Figure 9: Test result for Schwefel function with the initial position at $(-100.3, 25)$, $(-40.5, -45)$, $(17.8, 50.3)$, $(69.8, 70.6)$, and $(130.2, 98.7)$.

Table 9: Test results on Schwefel function (contour plot refer Figure 9)

Path No	Local minimums		Saddle
1	(-124.8170, 5.2615)	(-124.4369, 65.4794)	(-124.2794, 25.7779)
2	(5.2807, 65.4046)	(65.5185, 65.2612)	(26.0773, 65.3830)
3	(65.5185, 65.2612)	(203.7441, 65.5489)	(124.8765, 65.0411)
4	(-124.6262, 65.5132)	(-26.2200, 65.4095)	(-66.1930, 65.1174)
5	(-26.2200, 65.4095)	(5.2516, 65.4740)	(-3.9133, 65.3002)

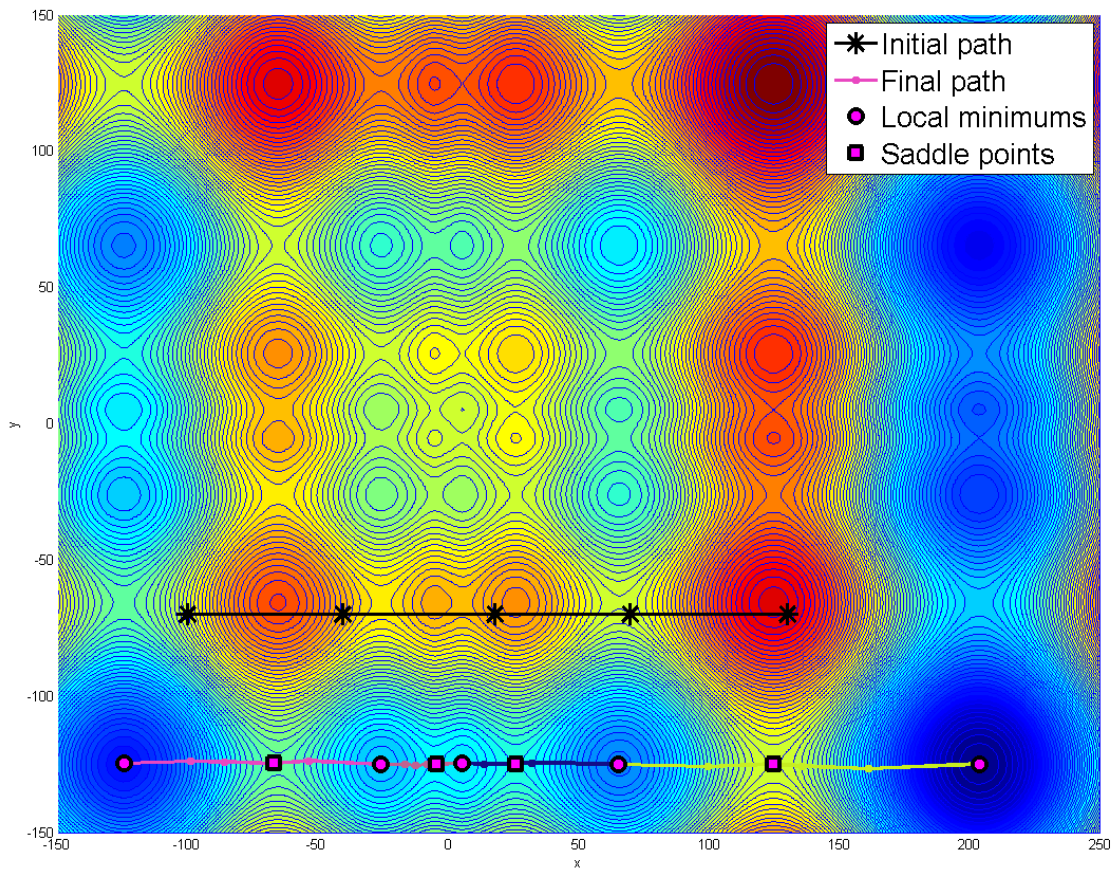


Figure 10: Test result for Schwefel function with the initial position at (-100.3, 25), (-40.5, 25), (17.8, 25), (69.8, 25), and (130.2, 25).

Table 10: Test results on Schwefel function (contour plot refer Figure 10)

Path No	Local minimums		Saddle
1	(-124.8274, -124.7472)	(-26.1493, -125.1159)	(-67.1757, -124.7392)
2	(-26.1493, -125.1159)	(5.2426, -124.6550)	(-4.6699, -124.8731)
3	(5.2402, -124.7391)	(65.3225, -124.8080)	(25.8758, -124.8290)
4	(65.4076, -125.0250)	(203.7606, -124.8008)	(124.7793, -124.8429)

4.5 Discussion

In sections 2.1 and 2.2, it is demonstrated with two examples that the method can locate multiple local minimums and saddle points. Here, we give a brief discussion on the rate of convergence of this method.

4.5.1 Convergence analysis

The convergence of the algorithm to the local minimum and the MEPs are discussed in this section. In this algorithm, the conjugate gradient method is adopted in the searching the local minimum. For a quadratic function with n variables, the method can guarantee the local minimum will be located in at most n iterations apart from round-off errors. The searching points converge to the local minimum quadratically. For a non-quadratic function with n variables, the searching process is usually iterative rather than n steps. The approximated conjugate directions generated using Eq.(3.3) are the directions corresponding to the local quadratic approximation to the non-quadratic function. The rate of convergence for the non-quadratic function depends on the response to changes in the local quadratic approximation from one iteration to another. When the searching point approaches the local minimum, it converges to the minimum quadratically. Hence, the choice of the initial position of the end points of the Bézier curve is very important. Those initial positions that require the least number of steps to converge to the bottom of the valley, where the local quadratic approximation is accurate, are the best choices for the initial position of the end control points. In addition, the line search method is employed to determine the step size in each conjugate direction. In each conjugate direction,

only several limited mini-steps are applied to locate the minimum along that direction in order to reduce the computational cost. In other words, the point we locate by the mini-steps search may not be the minimum point along that direction. This could expand the approximation error in terms of conjugacy. Ultimately it will lead to more steps for the conjugate gradient method to converge. There is a trade-off between the number of functional evaluations during the line search process in each conjugate directions and the number of iterations needed for the conjugate gradient method to converge to the minimum.

CHAPTER 5

SUMMARY AND FUTURE WORK

In this thesis, a global search algorithm named a concurrent multi transition pathways search algorithm is developed to locate several local minimums as well as saddle points simultaneously. No prior knowledge of the reactant and product is needed for our algorithm. Different from the current transition pathway and saddle point search methods, our algorithm can search multiple transition paths starting from one initial transition path, which can provide a better view of the PES than the methods that only search one transition path.

5.1 *Summary and Discussions*

The algorithm presented in this thesis includes two parts. One is the single transition pathway search, and the other is the multiple transition pathway search. The former part locates the local minimum as well as MEP for a single path. For the single transition pathway search, the conjugate gradient method is used to minimize the two end control points, which locates two local minimums. The intermediate control points are minimized along the conjugate directions in order to push the curve to MEP. A constrained degree elevation and reduction scheme for Bézier curve is developed to redistribute the intermediate control points. The output for the stage of single transition pathway search is the input for the stage of multiple transition pathway search. A curve subdivision scheme is developed so that multiple transition paths can be located. The algorithm is demonstrated by examples of LEPS potential, LEPS plus harmonic oscillator potential, and PESs defined by Rastrigin function and Schwefel function. Although the demonstration is conducted on two dimensional PESs, the algorithm works for higher dimensional systems as well.

The major advantage of the algorithm is that it is able to dynamically locate all the intermediate local minimums and saddle points between the reactant and product along one transition path. This provides a comprehensive view of the transition process along this particular path.

In this algorithm, Bézier curve used to represent the transition path can cause undesirable loops, especially at the two end positions. Loops should be avoided. The constrained degree

elevation and reduction of Bézier curve in Section 3.1.3 is introduced for this purpose. Previously, the author developed a degree reduction in the same way as the one described in Section 3.1.3 but only using two adjacent control points. The scheme works well if there is no abrupt change of potential energy along the path. However, loops can be introduced at the position where potential energy changes abruptly. In order to make the algorithm general, the author improved the scheme by introducing one more adjacent control point which is the one described in Section 3.1.3. It is much better in terms of eliminating loops. Yet, it has its own shortcoming. The smoothing scheme introduced in the degree reduction process prevents the curve from converging to a curved transition path. In order to solve this problem, the local degree elevation and reduction scheme is introduced as described in Section 3.1.4. In addition, during the implementation, the step size and the maximum number of steps for minimization along each conjugate direction are also adjusted to avoid loops. The step size for minimization of the intermediate control points is adaptive which includes the information of the potential energy changes on the surface. The maximum number minimization steps along each conjugate direction should not be too large. Otherwise, the loops will be introduced.

Another implementation issue is related to the stop criteria for searching the saddle point. During the climbing process, the control points with maximum energy on each sub curves climb up to locate the saddle points. The author first tried to eliminate the process of redistribution of control points during the climbing process. Unfortunately, loops can be formed at the positions with abrupt potential energy change. Then the redistribution process of control points is introduced to the climbing process. The issue of introducing the redistribution process is that the position of the control point with the maximum energy changes after redistributing the control points. The original control point with the maximum energy may not be the one with the maximum energy along the path after the degree elevation and reduction. Therefore, the algorithm needs to re-evaluate the energy level for the intermediate control points after redistributing the points in order to determine the next climbing point. This inevitably increases the computational cost. In addition, it is difficult to determine the stop criteria since the climbing points are changing from one iteration to another. It is not efficient to adopt the criteria related to the percentage change in potential energy at the point with the maximum energy. Although it will converge, it takes more iterations.

5.2 *Future work*

As mentioned in section 3.2.2, the curve subdivision scheme will treat some breakable curves as unbreakable ones in some rare cases. The corresponding remedy is also introduced in the same section. However, this remedy is not good enough, and it increases the computational cost. A better curve subdivision scheme could be developed as an extension of the current one.

In addition, for two particular stable states, the algorithm can only locate one transition path which consists of multiple curves with their end points connected together locating at multiple local minimums. There could be other possible paths between those two states. Figure 11 illustrates such cases, where A, B, C, D, and E are stable states. From the initial state A to the final state B, there are two possible transition paths. One is A-D-E-B, and the other is A-C-B. The algorithm can only locate either the transition path A-D-E-B or A-C-B. It is better to identify all the possible transition paths between the two states for the following two reasons. Firstly, identification of all the possible transition paths allows us to determine the correct activation energy. The algorithm only locates one transition path which could be the one with the energy barrier higher than the physical minimum energy barrier. This leads to an overestimation of the minimum energy barrier between the states A and B. For example, if the path A-C-B is the one with minimum energy barrier between states A and B. The algorithm may locate the path A-D-E-B instead of the path A-C-B. As a result, the minimum energy barrier between states A and B will be overestimated. Secondly, identification of all the possible transition paths provides us a better overview of the landscape of the PES. Therefore, the ideal case is to locate all the paths on the PES. Once the algorithm identified all the local minima as well as the corresponding saddle points, it becomes easy to identify the MEP from the reactant to the product. To locate all the transition paths on the PES, the algorithm needs several experiments with different initial guesses of the transition path.. The procedure is computationally expensive. Since the algorithm can identify multiple transition paths during one experiment, an experiment with different initial guesses may locate transition paths which are already identified by the previous experiments. In addition, for a big system with multiple atoms, it is difficult to obtain an analytical representation of the PES. In other words, we do not know the distribution of the saddle points and local minimums. As a result, it is difficult to develop criteria to determine whether all the transition paths are located or not for those experiments. To locate all the transition paths, we need a

systematic method. In the future, we will develop an algorithm that can locate as many transition paths on the PES as possible simultaneously, as an extension of the current method. To locate all the possible transition paths on the PES, the proposed method will incorporate a modified particle swarm optimization method.. The result of the proposed algorithm could give us a better overview of the landscapes of the PES. In addition, the proposed method is more efficient than the ones only searching individual ones.

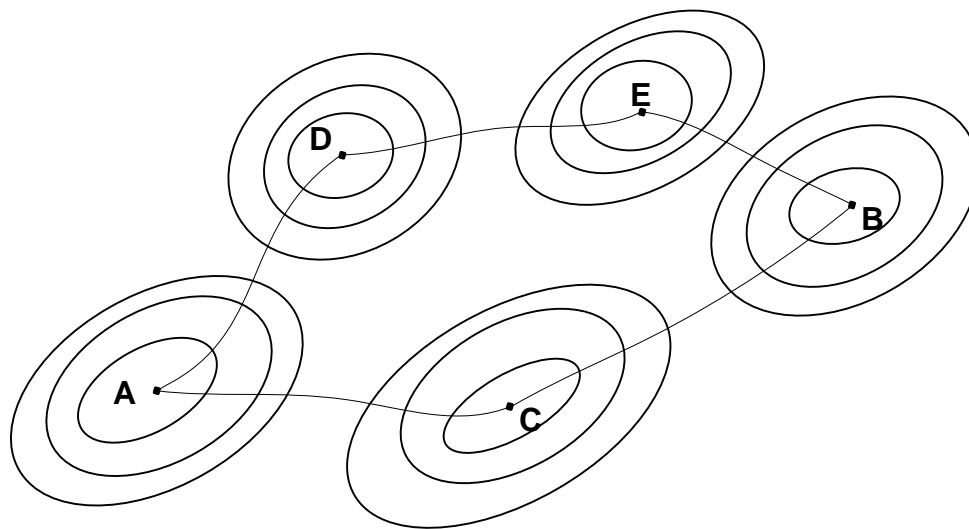


Figure 11: Illustration for two possible transition paths between two states

The ultimate goal of searching the saddle points is to determine the transition rates which are the key inputs for the phase transition simulation method. The accuracy of the KMC simulation is closely related to the accuracy of the transition rates and thus the energy value at the saddle points. Uncertainties are always involved in estimating the activation energy barrier thus the transition rates. The existing saddle point search methods are based on the information of available PESs which are constructed through first principles calculation. During the calculation, numerical setup and approximations for computability will inevitably introduce uncertainties such as the exchange-correlation treatment in the density functional theory (DFT). In addition, the transition rate estimated by transition state theory (TST) is an upper bound of the true but unknown rate. It is very important to take those uncertainties into account during

searching the activation energy barrier. Otherwise, the simulation results based on the inaccurate information of activation energy could be misleading. Another future work will focus on the uncertainty quantification related to the saddle point search.

APPENDIX

MATLAB SOURCE CODE FOR THE ALGORITHM

%This is the main source code for the algorithm. All the functions are listed in the same section.

```
clear all;
m=5;%m:# control points
A(1).L(1).p(:,1:m)=...
    input('initial position in column vector for control points');
n=size(A(1).L(1).p,2);%n: PES dimension
N_N=20; %N_N is the maximum number of iteration

h=0.0001;%h threshold for stop criteria
th=0.01;%th: threshold to determine if the two end points converges to the same point or not
N_inter=6;%N_inter: Maxi # of mini steps for intermediate control
    %points during the minimization process
N_climb=8;%N_climb: Maxi # of mini steps for intermediate control points
    %during climbing up process
N_end=20;%N_end: Maximum # of line mini step for end points during the
    %conjugate gradient minimization process
c3=1/95;%step size coefficients
c3_inter=1/50;%step size coefficients for intermediate control point
c3_climb=1/150;%step size coefficients during climbing process

c4=1/5;%percentage decrease in step size
c_deg=1/(3*m);
th_deg=c_deg*norm(A(1).L(1).p(:,end)-A(1).L(1).p(:,1));
%th_deg: threshold to determine whether two points are too close to each
%other during degree elevation and reduction

%Minimize the two end points to locate two local minimums
A(2).L(1).p= local_mini( A(1).L(1).p, N_N,N_inter,N_end,h,c3,c3_inter,c4);

j=1;%index for the # of paths we locate.
NP=5; %Minimum control points for those curves
i=2;%index for the total number of iterations in main code
```

```

k=1;
while 1
    i=i+1;
    SA=0; %number of lines in A(i+1).L
    for ii=1:size(A(i-1).L,2)
        b=size(A(i-1).L(ii).p,2);
        %make the curve to be fourth or fifth order before checking whether
        %there is extra local minimums
        if b<5
            for jj=b:4
                A(i-1).L(ii).p(:,1:(jj+1))=Be_degelevation(A(i-1).L(ii).p,jj,th_deg);
            end
        elseif b>6
            for jj=7:b
                A(i-1).L(ii).p(:,1:(jj-1))=Be_degredution(A(i-1).L(ii).p,jj,th_deg);
                A(i-1).L(ii).p(:,1:(jj))=[];
            end
        end
    end
end

for jj=1:size(A(i-1).L,2)
    B(1).L=curve_break(A(i-1).L(jj).p);
    SB=size(B(1).L,2); %number of newly produced curves in B(1).L
    if SB==1 %curve is unbreakable, then do not necessary to minimize
        %the end points
        %Redistribut the points through degelevation or degredution,
        %and then recheck if it is breakable or not.
        if size(B(1).L(1).p,2)==NP
            D(k).L(1).p(:,1:NP+1)=Be_degelevation(B(1).L(1).p,NP,th_deg);
        elseif size(B(1).L(1).p,2)==NP+1
            D(k).L(1).p(:,1:NP)=Be_degredution(B(1).L(1).p,NP+1,th_deg);
        end

        E(k).L=curve_break(D(k).L(1).p);
        SE=size(E(k).L,2);%SE:# of newly produced curves
        if SE==1 %the curve is still unbreakable
            C(1).L(j)=B(1).L; % C(1).L: unbreakable curves
        end
    end
end

```

```

j=j+1;
else
    %increase the control points of those new curve sections to
    %five if there are less then five
    for ii=1:SE
        if size(E(k).L(ii).p,2)<NP
            for iii=1:(NP-size(E(k).L(ii).p,2))
                sp=size(E(k).L(ii).p,2);
                E(k).L(ii).p(:,1:sp+1)=Be_degelevation(E(k).L(ii).p,sp,th_deg);
            end
        end
    end
    %Minimize the end points for those new curves in order to
    %locate the new local minimums
    for kk=1:SE
        E(k).L(kk).p=local_mini(E(k).L(kk).p,N_N,N_inter,N_end,h,c3,c3_inter,c4);
    end
    %check if there is loop (two end points converges to the
    %same point).If yes, means the curve is actually not breakable
    kk=1;
    while kk<=SE
        if norm(E(k).L(kk).p(:,1)-E(k).L(kk).p(:,end))/norm(A(1).L(1).p(:,1)-A(1).L(1).p(:,end))<th
            C(1).L(j)=B(1).L;
            j=j+1;
            break
        else
            if kk==SE
                A(i).L(SA+1:(SA+SE))=E(k).L;
                SA=size(A(i).L,2);
            end
            kk=kk+1;
        end
    end
    k=k+1;
else
    %increase the control points of those new curve sections to

```

```

%five if there are less then five
for ii=1:SB
if size(B(1).L(ii).p,2)<NP
for iii=1:(NP-size(B(1).L(ii).p,2))
sp=size(B(1).L(ii).p,2);
B(1).L(ii).p(:,1:sp+1)=Be_degelevation(B(1).L(ii).p,sp,th_deg);
end
end
end
%Minimize the end points for those new curves in order to
%locate the new local minimums
for kk=1:SB
B(1).L(kk).p=local_mini(B(1).L(kk).p,N_N,N_inter,N_end,h,c3,c3_inter,c4);
end
%check if there is loop (two end points converges to the
%same point). If yes, means the curve is actually not breakable
kk=1;
while kk<=SB
if norm(B(1).L(kk).p(:,1)-B(1).L(kk).p(:,end))/norm(A(1).L(1).p(:,1)-A(1).L(1).p(:,end))<th
if kk==1
C(1).L(j)=B(1).L(SB);
j=j+1;
else
C(1).L(j)=B(1).L(1);
j=j+1;
end
break
else
if kk==SB
A(i).L(SA+1:(SA+SB))=B(1).L;
SA=size(A(i).L,2);
end
kk=kk+1;
end
end
end
end

```



```

end

if SA==0;%check if there is newly produced curve section
    break
end
end

%check if two curve are overlapped
k=1;
M=[];
for ii=1:size(C(1).L,2)-1
    for jj=ii+1:size(C(1).L,2)
        if norm(C(1).L(ii).p(:,1)-C(1).L(jj).p(:,1))/norm(A(1).L(1).p(:,1)-A(1).L(1).p(:,end))<th
            if norm(C(1).L(ii).p(:,end)-C(1).L(jj).p(:,end))/norm(A(1).L(1).p(:,1)-A(1).L(1).p(:,end))<th
                M(k)=ii;
                k=k+1;
            end
        elseif norm(C(1).L(ii).p(:,1)-C(1).L(jj).p(:,end))/norm(A(1).L(1).p(:,1)-A(1).L(1).p(:,end))<th
            if norm(C(1).L(ii).p(:,end)-C(1).L(jj).p(:,1))/norm(A(1).L(1).p(:,1)-A(1).L(1).p(:,end))<th
                M(k)=ii;
                k=k+1;
            end
        end
    end
end
end
if size(M,2)>0
    for ii=1:size(M,2)
        C(1).L(M(ii)).p=[];
    end
end

%Locate the saddle points for each curve.
for ii=1:size(C(1).L,2)
    N(1).L(ii).p=C(1).L(ii).p;
    C(1).L(ii).p=saddle_search(C(1).L(ii).p,c3_climb,c4,n,N_climb);
end

%Output all the local minimums.

```

```

for ii=1:j-1
    disp(C(1).L(ii).p(:,1));
    disp(C(1).L(ii).p(:,end));
end

```

*%This function minimize the two end points in order to locate two local
%minimums. Also the intermediate control points move along the
%conjugate directions which will gradually converge to MEP.*

```

function H = local_mini( p, N_N,N_inter,N_end,h,c3,c3_inter,c4)

```

```

%a_a_*: index for the # of degree elevation and reduction

```

```

a_a_for=1; %index for first half

```

```

a_a_back=1;%index for second half

```

```

a_a_tot=1;%index for the whold curve

```

```

m_=size(p,2);

```

```

c_deg_=1/(3*m_);

```

```

th_deg_=c_deg_*norm(p(:,end)-p(:,1));

```

```

A_A(1).p=p;%A_A(i_i).p: control points matrix

```

```

n_n=size(p,1);%dimension of the PES

```

```

A_A(2).p=ones(n_n,size(A_A(1).p,2));

```

```

%Minimize the two end points using conjugate gradient method

```

```

A_A(2).p(:,1)=conjugate_mini(A_A(1).p(:,1),c3,c4,N_end);

```

```

A_A(2).p(:,end)=conjugate_mini(A_A(1).p(:,end),c3,c4,N_end);

```

```

%Minimize all the intermediate control points along the conjugate

```

```

%directions with positive eigenvalues

```

```

for ii_ii=2:ceil(size(A_A(2).p,2)/2)

```

```

    A_A(2).p(:,ii_ii)=inter_mini(A_A(2).p(:,ii_ii-1),A_A(1).p(:,ii_ii),c3_inter,c4,n_n,N_inter);

```

```

end

```

```

ii_ii=size(A_A(2).p,2)-1;

```

```

while ii_ii>ceil(size(A_A(2).p,2)/2)

```

```

    A_A(2).p(:,ii_ii)=inter_mini(A_A(2).p(:,ii_ii+1),A_A(1).p(:,ii_ii),c3_inter,c4,n_n,N_inter);

```

```

    ii_ii=ii_ii-1;

```

```

end

```

```

%degree elevation
B_B(2).p=A_A(2).p;
b_b=size(A_A(2).p,2);
c_c=index_loop(A_A(2).p);
if size(c_c,2)>0
    if max(c_c)<ceil(m_/2)
        C_C(1).p(:,1:ceil(b_b/2)+1)=Be_degelevation(A_A(2).p(:,1:ceil(b_b/2)),ceil(b_b/2),th_deg_);
        A_A(2).p(:,1:ceil(b_b/2)+1)=C_C(1).p;
        A_A(2).p(:,ceil(b_b/2)+2:b_b+1)=B_B(2).p(:,ceil(b_b/2)+1:end);
        a_a_for=0;
    elseif min(c_c)>ceil(m_/2)
        C_C(1).p(:,1:b_b-ceil(b_b/2)+2)=Be_degelevation(...
            A_A(2).p(:,ceil(b_b/2):end),b_b-ceil(b_b/2)+1,th_deg_);
        A_A(2).p(:,ceil(b_b/2):b_b+1)=C_C(1).p;
        a_a_back=0;
    else
        A_A(2).p(:,1:(b_b+1))=Be_degelevation(A_A(2).p,b_b,th_deg_);
        a_a_tot=0;
    end
end
i_j=2; %index for the total # of iterations
j_j=2;
while i_i<N_N%N_N Maximum number of iteration to search local minimum
    if abs((fun_value(A_A(i_i).p(:,1))-fun_value(A_A(i_i-1).p(:,1)))/fun_value(A_A(i_i-1).p(:,1)))>h&&...
        abs((fun_value(A_A(i_i).p(:,end))-fun_value(A_A(i_i-1).p(:,end)))/fun_value(A_A(i_i-1).p(:,end))))>h

        i_i=i_i+1;
        A_A(i_i).p=ones(n_n,size(A_A(i_i-1).p,2));

        A_A(i_i).p(:,1)=conjugate_mini(A_A(i_i-1).p(:,1),c3,c4,N_end);
        A_A(i_i).p(:,end)=conjugate_mini(A_A(i_i-1).p(:,end),c3,c4,N_end);

        for ii_ii=2:ceil(size(A_A(i_i).p,2)/2)
            A_A(i_i).p(:,ii_ii)=inter_mini(A_A(i_i-1).p(:,ii_ii-1),...
                A_A(i_i-1).p(:,ii_ii),c3_inter,c4,n_n,N_inter);

```

```

end

ii_ii=size(A_A(i_i).p,2)-1;
while ii_ii>ceil(size(A_A(i_i).p,2)/2)
    A_A(i_i).p(:,ii_ii)=inter_mini(A_A(i_i-1).p(:,ii_ii+1),...
        A_A(i_i-1).p(:,ii_ii),c3_inter,c4,n_n,N_inter);
    ii_ii=ii_ii-1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%redistribute the control points using degree elevation and reduction scheme
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

b_b=size(A_A(i_i).p,2);
B_B(i_i).p=A_A(i_i).p;
c_c=index_loop(A_A(i_i).p);
if size(c_c,2)>0
    if max(c_c)<ceil(m_/2)
        if a_a_for==0
            C_C(j_j).p(:,1:ceil(b_b/2)-1)=Be_degreduction(...
                A_A(i_i).p(:,1:ceil(b_b/2)),ceil(b_b/2),th_deg_);
            A_A(i_i).p(:,1:ceil(b_b/2)-1)=C_C(j_j).p;
            A_A(i_i).p(:,ceil(b_b/2):b_b-1)=B_B(i_i).p(:,ceil(b_b/2)+1:end);
            A_A(i_i).p(:,end)=[];
            j_j=j_j+1;
            a_a_for=a_a_for+1;
        else
            C_C(j_j).p(:,1:ceil(b_b/2)+1)=Be_degelevation(...
                A_A(i_i).p(:,1:ceil(b_b/2)),ceil(b_b/2),th_deg_);
            A_A(i_i).p(:,1:ceil(b_b/2)+1)=C_C(j_j).p;
            A_A(i_i).p(:,ceil(b_b/2)+2:b_b+1)=B_B(i_i).p(:,ceil(b_b/2)+1:end);
            j_j=j_j+1;
            a_a_for=0;
        end
    elseif min(c_c)>ceil(m_/2)
        if a_a_back==0
            C_C(j_j).p(:,1:b_b-ceil(b_b/2))=Be_degreduction(...
                A_A(i_i).p(:,ceil(b_b/2):end),b_b-ceil(b_b/2)+1,th_deg_);
            A_A(i_i).p(:,ceil(b_b/2):b_b-1)=C_C(j_j).p;
            A_A(i_i).p(:,end)=[];

```

```

j_j=j_j+1;
a_a_back=a_a_back+1;
else
C_C(j_j).p(:,1:b_b-ceil(b_b/2)+2)=Be_degelevation(...
A_A(i_i).p(:,ceil(b_b/2):end),b_b-ceil(b_b/2)+1,th_deg_);
A_A(i_i).p(:,ceil(b_b/2):b_b+1)=C_C(j_j).p;
j_j=j_j+1;
a_a_back=0;
end
else
if a_a_tot==0
A_A(i_i).p(:,1:(b_b-1))=Be_degreduction(A_A(i_i).p,b_b,th_deg_);
A_A(i_i).p(:,end)=[];
a_a_tot=a_a_tot+1;
else
A_A(i_i).p(:,1:(b_b+1))=Be_degelevation(A_A(i_i).p,b_b,th_deg_);
a_a_tot=0;
end
end
end
end

%%%%%%%%%%
elseif abs((fun_value(A_A(i_i).p(:,1))-fun_value(A_A(i_i-1).p(:,1)))/fun_value(A_A(i_i-1).p(:,1)))<h&&...
abs((fun_value(A_A(i_i).p(:,end))-fun_value(A_A(i_i-1).p(:,end)))/fun_value(A_A(i_i-1).p(:,end)))>h
i_i=i_i+1;
A_A(i_i).p=ones(n_n,size(A_A(i_i-1).p,2));
A_A(i_i).p(:,1)=A_A(i_i-1).p(:,1);
A_A(i_i).p(:,end)=conjugate_mini(A_A(i_i-1).p(:,end),c3,c4,N_end);

for ii_ii=2:ceil(size(A_A(i_i).p,2)/2)
A_A(i_i).p(:,ii_ii)=inter_mini(A_A(i_i-1).p(:,ii_ii-1),...
A_A(i_i-1).p(:,ii_ii),c3_inter,c4,n_n,N_inter);
end
ii_ii=size(A_A(i_i).p,2)-1;
while ii_ii>ceil(size(A_A(i_i).p,2)/2)
A_A(i_i).p(:,ii_ii)=inter_mini(A_A(i_i-1).p(:,ii_ii+1),...
A_A(i_i-1).p(:,ii_ii),c3_inter,c4,n_n,N_inter);

```

```

    ii_ii=ii_ii-1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
b_b=size(A_A(i_i).p,2);
B_B(i_i).p=A_A(i_i).p;
c_c=index_loop(A_A(i_i).p);
if size(c_c,2)>0
    if max(c_c)<ceil(m_/2)
        if a_a_for==0
            C_C(j_j).p(:,1:ceil(b_b/2)-1)=Be_degreduction(...
                A_A(i_i).p(:,1:ceil(b_b/2)),ceil(b_b/2),th_deg_);
            A_A(i_i).p(:,1:ceil(b_b/2)-1)=C_C(j_j).p;
            A_A(i_i).p(:,ceil(b_b/2):b_b-1)=B_B(i_i).p(:,ceil(b_b/2)+1:end);
            A_A(i_i).p(:,end)=[];
            j_j=j_j+1;
            a_a_for=a_a_for+1;
        else
            C_C(j_j).p(:,1:ceil(b_b/2)+1)=Be_degelevation(...
                A_A(i_i).p(:,1:ceil(b_b/2)),ceil(b_b/2),th_deg_);
            A_A(i_i).p(:,1:ceil(b_b/2)+1)=C_C(j_j).p;
            A_A(i_i).p(:,ceil(b_b/2)+2:b_b+1)=B_B(i_i).p(:,ceil(b_b/2)+1:end);
            j_j=j_j+1;
            a_a_for=0;
        end
    elseif min(c_c)>ceil(m_/2)
        if a_a_back==0
            C_C(j_j).p(:,1:b_b-ceil(b_b/2))=Be_degreduction(...
                A_A(i_i).p(:,ceil(b_b/2):end),b_b-ceil(b_b/2)+1,th_deg_);
            A_A(i_i).p(:,ceil(b_b/2):b_b-1)=C_C(j_j).p;
            A_A(i_i).p(:,end)=[];
            j_j=j_j+1;
            a_a_back=a_a_back+1;
        else
            C_C(j_j).p(:,1:b_b-ceil(b_b/2)+2)=Be_degelevation(...
                A_A(i_i).p(:,ceil(b_b/2):end),b_b-ceil(b_b/2)+1,th_deg_);

```

```

    A_A(i_i).p(:,ceil(b_b/2):b_b+1)=C_C(j_j).p;
    j_j=j_j+1;
    a_a_back=0;
end
else
    if a_a_tot==0
        A_A(i_i).p(:,1:(b_b-1))=Be_degreduction(A_A(i_i).p,b_b,th_deg_);
        A_A(i_i).p(:,end)=[];
        a_a_tot=a_a_tot+1;
    else
        A_A(i_i).p(:,1:(b_b+1))=Be_degelevation(A_A(i_i).p,b_b,th_deg_);
        a_a_tot=0;
    end
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elseif abs((fun_value(A_A(i_i).p(:,1))-fun_value(A_A(i_i-1).p(:,1)))/fun_value(A_A(i_i-1).p(:,1)))>h&&...
    abs((fun_value(A_A(i_i).p(:,end))-fun_value(A_A(i_i-1).p(:,end)))/fun_value(A_A(i_i-
1).p(:,end)))<h
    i_i=i_i+1;
    G(:,2*(i_i-2)+1)=-grad(A_A(i_i-1).p(:,1));
    A_A(i_i).p=ones(n_n,size(A_A(i_i-1).p,2));
    A_A(i_i).p(:,1)=conjugate_mini(A_A(i_i-1).p(:,1),c3,c4,N_end);
    A_A(i_i).p(:,end)= A_A(i_i-1).p(:,end);

    for ii_ii=2:ceil(size(A_A(i_i).p,2)/2)
        A_A(i_i).p(:,ii_ii)=inter_mini(A_A(i_i-1).p(:,ii_ii-1),...
            A_A(i_i-1).p(:,ii_ii),c3_inter,c4,n_n,N_inter);
    end
    ii_ii=size(A_A(i_i).p,2)-1;
    while ii_ii>ceil(size(A_A(i_i).p,2)/2)
        A_A(i_i).p(:,ii_ii)=inter_mini(A_A(i_i-1).p(:,ii_ii+1),...
            A_A(i_i-1).p(:,ii_ii),c3_inter,c4,n_n,N_inter);
        ii_ii=ii_ii-1;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
b_b=size(A_A(i_i).p,2);

```

```

B_B(i_i).p=A_A(i_i).p;
c_c=index_loop(A_A(i_i).p);
if size(c_c,2)>0
    if max(c_c)<ceil(m_/2)
        if a_a_for==0
            C_C(j_j).p(:,1:ceil(b_b/2)-1)=Be_degreduction(...
                A_A(i_i).p(:,1:ceil(b_b/2)),ceil(b_b/2),th_deg_);
            A_A(i_i).p(:,1:ceil(b_b/2)-1)=C_C(j_j).p;
            A_A(i_i).p(:,ceil(b_b/2):b_b-1)=B_B(i_i).p(:,ceil(b_b/2)+1:end);
            A_A(i_i).p(:,end)=[];
            j_j=j_j+1;
            a_a_for=a_a_for+1;
        else
            C_C(j_j).p(:,1:ceil(b_b/2)+1)=Be_degelevation(...
                A_A(i_i).p(:,1:ceil(b_b/2)),ceil(b_b/2),th_deg_);
            A_A(i_i).p(:,1:ceil(b_b/2)+1)=C_C(j_j).p;
            A_A(i_i).p(:,ceil(b_b/2)+2:b_b+1)=B_B(i_i).p(:,ceil(b_b/2)+1:end);
            j_j=j_j+1;
            a_a_for=0;
        end
    elseif min(c_c)>ceil(m_/2)
        if a_a_back==0
            C_C(j_j).p(:,1:b_b-ceil(b_b/2))=Be_degreduction(...
                A_A(i_i).p(:,ceil(b_b/2):end),b_b-ceil(b_b/2)+1,th_deg_);
            A_A(i_i).p(:,ceil(b_b/2):b_b-1)=C_C(j_j).p;
            A_A(i_i).p(:,end)=[];
            j_j=j_j+1;
            a_a_back=a_a_back+1;
        else
            C_C(j_j).p(:,1:b_b-ceil(b_b/2)+2)=Be_degelevation(...
                A_A(i_i).p(:,ceil(b_b/2):end),b_b-ceil(b_b/2)+1,th_deg_);
            A_A(i_i).p(:,ceil(b_b/2):b_b+1)=C_C(j_j).p;
            j_j=j_j+1;
            a_a_back=0;
        end
    end
else
    if a_a_tot==0

```



```

        A_A(i_i).p(:,1:(b_b-1))=Be_degreduction(A_A(i_i).p,b_b,th_deg_);
        A_A(i_i).p(:,end)=[];
        a_a_tot=a_a_tot+1;
    else
        A_A(i_i).p(:,1:(b_b+1))=Be_degelevation(A_A(i_i).p,b_b,th_deg_);
        a_a_tot=0;
    end
end
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else
    break
end
end
end
H=A_A(i_i).p;
end

```

%This function minimize the two end points using conjugate gradient method

```

function H = conjugate_mini( p ,c3,c4,N_end)
n_n=size(p,1);
p_p=ones(n_n,n_n);
s_s=ones(n_n,n_n); %s_s: conjugate gradient search directions
G=ones(n_n,2); %gradients
G(:,1)=-grad(p)';
s_s(:,1)=G(:,1);
%Minimize the end control points along s_s(:,1) direction (inexact line search)
p_p(:,1)=line_mini(p,s_s(:,1),c3,c4,N_end);
%Minimize the end control points along the conjugate directions iteratively
for i_i=2:n_n
    G(:,i_i)=-grad(p_p(:,i_i-1));
    s_s(:,i_i)=G(:,i_i)+dot(G(:,i_i),G(:,i_i))/dot(G(:,i_i-1),G(:,i_i-1))*s_s(:,i_i-1);
    p_p(:,i_i)=line_mini(p_p(:,i_i-1),s_s(:,i_i),c3,c4,N_end);
end
H=p_p(:,i_i);

```

end

***%This function does line minimization to two end points along conjugate
%gradient directions***

```
function H = line_mini( p1,s,c3,c4,N1_N1 )
%N1_N1:# of mini steps
h_h=0.000001;%threshold for stop criteria
h_h1=-0.001;%threshold when will decrease the step size
c3_c3=c3;
c4_c4=c4;
n_n=size(p1,1);
p1_p1=ones(n_n,2);
p1_p1(:,1)=p1;
G(:,1)=-grad(p1_p1(:,1));
ii=1;
while ii<N1_N1&&abs(dot(G(:,1),s)/norm(s))>h_h
    ii=ii+1;
    p1_p1(:,2)=p1_p1(:,1)+c3_c3*dot(G(:,1),s)/norm(s)^2*s;
    G(:,2)=-grad(p1_p1(:,2));
    %when there is change in direction of the gradient, decrease the step size
    if dot(G(:,1),G(:,2))/norm(G(:,1))/norm(G(:,2))<h_h1
        c3_c3=c4_c4*c3_c3;
    end
    p1_p1(:,1)=p1_p1(:,2);
    G(:,1)=G(:,2);
end
H=p1_p1(:,1);
end
```

***%This function minimize the intermediate control points along the conjugate
%direction with positive eigenvalues***

```
function H = inter_mini( p1,p2,c3,c4,n,MAX_N )
%p2 is the point which will be minimized
%p1 is fixed, only to determine the maximization direction
%MAX_N:#of mini-step in conjugate direction
```

```

c5_c5=1/2;
b1=0.001;%threshold for convergence criteria
b2=0.000001;%threshold for convergence criteria
h_h1=0;
x1=ones(n,n);%x1: matrix to store the positions of p2 iteratively
%g0:gradient at p2, g1 is a n by(n-1)matrix to store the gradient g1,g2...
%in Beal's formula
g1=ones(n,n-1);
s=ones(n,n);
s(:,1)=p2-p1;%direction with negative eigenvalue(maximization direction)
%Locate the maximum point along s0=s(:,1) in order to determine g1 in Beal's formula
x1(:,1)=p2;
xmid=1/2*(p1+p2);
g0(:,1)=grad(xmid);
ii=1;
X1(:,1)=1/2*(p1+p2);
G1(:,1)=g0(:,1);
c3_c3=c3;

while ii<MAX_N&&abs(dot(G1(:,ii),s(:,1))/norm(s(:,1)))>b2
    ii=ii+1;
    X1(:,ii)=X1(:,ii-1)+c3_c3*dot(G1(:,ii-1),s(:,1))/norm(s(:,1))^2*s(:,1);
    G1(:,ii)=grad(X1(:,ii));
    if dot(G1(:,ii-1),s(:,1))*dot(G1(:,ii),s(:,1))/norm(s(:,1))^2<h_h1
        c3_c3=c4*c3_c3;
    end
end
x1(:,1)=X1(:,ii); %maximum point along the s0
g1(:,1)=grad(x1(:,1));

%calculate s(:,2)represents s1 in Beal's formula (first conjugate direction
%with positive eigenvalue)
if abs(dot(s(:,1),(g1(:,1)-g0(:,1))))>b1
    s(:,2)=-g1(:,1)+dot(g1(:,1),(g1(:,1)-g0(:,1)))/...
        dot(s(:,1),(g1(:,1)-g0(:,1)))*s(:,1);
else
    s(:,2)=-g1(:,1)+dot(g1(:,1),(g1(:,1)-g0(:,1)))*s(:,1);

```

end

%minimize p2 along s(:,2)

ii=1;

X1(:,1)=p2;

G1(:,1)=grad(X1(:,1));

c3_c3=c3;

while ii<MAX_N&&abs(dot(G1(:,ii),s(:,2))/norm(s(:,2))/norm(G1(:,ii))*norm(G1(:,ii)))>b2

ii=ii+1;

if abs(dot(G1(:,ii-1),s(:,2))/norm(s(:,2))/norm(G1(:,ii-1)))<0.5

X1(:,ii)=X1(:,ii-1)+c3_c3*dot(-G1(:,ii-1),s(:,2))/norm(s(:,2))^2*s(:,2);

G1(:,ii)=grad(X1(:,ii));

if dot(G1(:,ii-1),s(:,1))*dot(G1(:,ii),s(:,1))/norm(s(:,1))^2<h_h1

c3_c3=c4*c3_c3;

end

else

X1(:,ii)=X1(:,ii-1)+c5_c5*c3_c3*dot(-G1(:,ii-1),s(:,2))/norm(s(:,2))^2*s(:,2);

G1(:,ii)=grad(X1(:,ii));

if dot(G1(:,ii-1),s(:,1))*dot(G1(:,ii),s(:,1))/norm(s(:,1))^2<h_h1

c3_c3=c4*c3_c3;

end

end

end

x1(:,2)=X1(:,ii);*%new position for p2 along s(:,2)*

g1(:,2)=grad(x1(:,2));

%if the dimension of the PES is larger than two, then minimize p2 along all the conjugate directions

%with negative eigenvalues iteratively

jj=3;

while jj<=n

if abs(dot(s(:,1),g1(:,1)-g0(:,1)))>b1

s(:,jj)=-g1(:,jj-1)+dot(g1(:,jj-1),g1(:,1)-g0(:,1))/...

dot(s(:,1),g1(:,1)-g0(:,1))*s(:,1)+dot(g1(:,jj-1),g1(:,jj-1))/...

dot(g1(:,jj-2),g1(:,jj-2))*s(:,jj-1);

else

s(:,jj)=-g1(:,jj-1)+dot(g1(:,jj-1),g1(:,jj-1))/...

dot(g1(:,jj-2),g1(:,jj-2))*s(:,jj-1);

```

end

%minimize p2 along s(:,jj) direction
ii=1;
X1(:,1)=x1(:,jj-1);
G1(:,1)=g1(:,jj-1);
c3_c3=c3;
while ii<MAX_N&&abs(dot(G1(:,ii),s(:,2))/norm(s(:,2))/norm(G1(:,ii))*norm(G1(:,ii)))>b2
    ii=ii+1;
    if abs(dot(G1(:,ii-1),s(:,2))/norm(s(:,2))/norm(G1(:,ii-1)))<0.5
        X1(:,ii)=X1(:,ii-1)+c3_c3*dot(-G1(:,ii-1),s(:,2))/norm(s(:,2))^2*s(:,2);
        G1(:,ii)=grad(X1(:,ii));
        if dot(G1(:,ii-1),s(:,1))*dot(G1(:,ii),s(:,1))/norm(s(:,1))^2<h_h1
            c3_c3=c4*c3_c3;
        end
    else
        X1(:,ii)=X1(:,ii-1)+c5_c5*c3_c3*dot(-G1(:,ii-1),s(:,2))/norm(s(:,2))^2*s(:,2);
        G1(:,ii)=grad(X1(:,ii));
        if dot(G1(:,ii-1),s(:,1))*dot(G1(:,ii),s(:,1))/norm(s(:,1))^2<h_h1
            c3_c3=c4*c3_c3;
        end
    end
end
end
end
x1(:,jj)=X1(:,ii);
g1(:,jj)=grad(x1(:,jj));
jj=jj+1;
end
H=x1(:,jj-1);
End

```

%This function is used to determine if there is zigzag along the path and the position of the %zigzag

```

function H = index_loop( p )
jj=0;
for ii=3:size(p,2)
    if dot(p(:,ii)-p(:,ii-1),p(:,ii-1)-p(:,ii-2))/norm(p(:,ii)-p(:,ii-1))/norm(p(:,ii-1)-p(:,ii-2))<0.99

```

```

    jj=jj+2;
    index(jj-1:jj)=[ii-2, ii];
end
end
if jj>0
    H=index;
else
    H=[];
end
end
end

```

%This function elevates the degree of the curve by one

```

function H =Be_degelevation( p,m,th_deg)
% m:# of control point before degree elevation
% q: Coordinate matrix for the elevated m+1 control points
% p: Coordinate matrix for the original m control points
% th_deg: threshold to determine if two point are too close
q(:,1)=p(:,1); %the two end points are fixed during elevation process
q(:,m+1)=p(:,m);
h_h=1/3;
%if two points are too close, then the new points is set as the arithmetic
%average
for i=1:m-1
    q(:,i+1)=i/(m)*p(:,i)+(1-i/(m))*p(:,i+1);
    if norm(q(:,i+1)-(p(:,i+1)+p(:,i))/2)>h_h*norm(p(:,i+1)-p(:,i)) ,
        q(:,i+1)=(p(:,i+1)+p(:,i))/2;
    end
end
for i=2:m
    if norm(q(:,i)-q(:,i-1))<th_deg
        q(:,i)=(q(:,i+1)+q(:,i-1))/2;
    end
end
%check the distance between two points, if too close, reset as middle position
if norm(q(:,m+1)-q(:,m))<th_deg
    q(:,m)=(q(:,m+1)+q(:,m-1))/2;

```

```

end
H=q;
end

```

%This function reduces the degree of the curve by one

```

function H =Be_degreduction( p,m,th_deg )
%#m:# of control point before degree reduction
%n_n:dimension for the PES
%q: Coordinate matrix for the reduced m-1 control points
%p:Coordinate matrix for the original m control points
%q1&q2: Coordinate matrix for the reduced control points from forward &
%backward procedure respectively.
%th_deg:threshold to determine if two point are too close
n_n=size(p,1);
q1=ones(n_n,m-2);
q2=ones(n_n,m-1);
q(:,1)=p(:,1); %the two end points are fixed for degree reduction
q(:,m-1)=p(:,m);

%points generated by using the information of two adjacent control points
q1_1=ones(n_n,m-3);
q1_2=ones(n_n,m-3);
q1_3=ones(n_n,m-3);
q2_1=ones(n_n,m-3);
q2_2=ones(n_n,m-3);
q2_3=ones(n_n,m-3);
%generate the control points forwardly
q1(:,1)=p(:,1);
for i=2:m-2
    q1_1(:,i-1)=(m*p(:,i)-(i-1)*q1(:,i-1))/(m+1-i);
    q1_2(:,i-1)=(m*p(:,i+1)-(i-1)*p(:,i))/(m+1-i);
    q1_3(:,i-1)=(m*p(:,i+2)-(i-1)*p(:,i+1))/(m+1-i);
    q1(:,i)=(q1_1(:,i-1)+q1_2(:,i-1)+q1_3(:,i-1))/3;
end
%generate the control points backwardly
q2(:,m-1)=p(:,m);

```

```

i=m-1;
while i>=3
    q2_1(:,i-2)=(m*p(:,i)-(m-i)*q2(:,i))/i;
    q2_2(:,i-2)=(m*p(:,i-1)-(m-i)*p(:,i))/i;
    q2_3(:,i-2)=(m*p(:,i-2)-(m-i)*p(:,i-1))/i;
    q2(:,i-1)=(q2_1(:,i-2)+q2_2(:,i-2)+q2_3(:,i-2))/3;
    i=i-1;
end
%generate the final control points by linearly interpolating the two set of
%control points obtained from the forward and backward procedure.
for i=2:m-2
    w=i/m;
    q(:,i)=(1-w)*q1(:,i)+w*q2(:,i);
end

for i=2:m-2
    if norm(q(:,i)-q(:,i-1))<th_deg
        q(:,i)=(q(:,i+1)+q(:,i-1))/2;
    end
end
%check the distance between two points, if too close, reset as middle
%position
if norm(q(:,m-1)-q(:,m-2))<th_deg
    q(:,m-2)=(q(:,m-1)+q(:,m-3))/2;
end
H=q;
end

```

%This function calculates the function value at position p

```

function H= fun_value(p)
a=sym('A',[1 size(p,1)]);
z=ObjectiveF(a);
for i=1:size(p,1)
    z=subs(z,{a(i)},{p(i,1)});
end
H=z;

```


end

%This function calculates the gradient at p

```
function H= grad(p)
a=sym('A',[1 size(p,1)]);
z=ObjectiveF(a);
L =jacobian(z,a);
for i=1:size(p,1)
    L=subs(L,{a(i)},{p(i,1)});
end
H=L;
end
```

%This function serves as the input of the objective functions

```
function H = ObjectiveF( p )
%Objective funtion (Rastrigin)
H= 20+p(1).^2-10.*cos(2*pi.*p(1))+p(2).^2-10.*cos(2*pi.*p(2));
%Objective funtion (Schwefel)
H= 418.9829*2-p(1).*sin(sqrt(abs(p(1))))-p(2).*sin(sqrt(abs(p(2))));
end
```

*%This function breaks one curve into two curve sections if the curve is
%breakable. This function works only when the input curve has five or six
%control points. Curves with other number of control points will be treated
%as invalid input*

```
function H = curve_break( p )
h2_h2=-0.05;
NEWCURVE=0;%# of newly produced curves
if size(p,2)==5
    G_G=ones(size(p,1),5);
    for ii_ii=1:3
        G_G(:,ii_ii)=-grad(p(:,ii_ii+1)); %Calculate the gradient at the
    end %intermediate control points
    if dot(p(:,1)-p(:,2),G_G(:,1))/norm(p(:,1)-p(:,2))/norm(G_G(:,1))...
```

```

    <h2_h2&&dot(p(:,5)-p(:,4),G_G(:,3))/norm(p(:,5)-p(:,4))/...
    norm(G_G(:,3))<h2_h2
A_A(1).L(1).p=p(:,1:3);
A_A(1).L(2).p=p(:,3:5);
NEWCURVE=NEWCURVE+1;
elseif dot(p(:,1)-p(:,2),G_G(:,1))/norm(p(:,1)-p(:,2))/norm(G_G(:,1))...
    <h2_h2&&dot(p(:,5)-p(:,4),G_G(:,3))/norm(p(:,5)-p(:,4))/...
    norm(G_G(:,3))>h2_h2
if dot(p(:,3)-p(:,2),G_G(:,2))/norm(p(:,3)-p(:,2))/...
    norm(G_G(:,2))<h2_h2
    A_A(1).L(1).p=p(:,1:3);
    A_A(1).L(2).p=p(:,3:5);
    NEWCURVE=NEWCURVE+1;
else
    if fun_value(p(:,2))>fun_value(p(:,3))&&fun_value(p(:,3))>...
        fun_value(p(:,4))
    else
        A_A(1).L(1).p=p(:,1:2);
        A_A(1).L(2).p=p(:,2:5);
        NEWCURVE=NEWCURVE+1;
    end
end
elseif dot(p(:,1)-p(:,2),G_G(:,1))/norm(p(:,1)-p(:,2))/...
    norm(G_G(:,1))>h2_h2&&dot(p(:,5)-p(:,4),G_G(:,3))/...
    norm(p(:,5)-p(:,4))/norm(G_G(:,3))<h2_h2
if dot(p(:,3)-p(:,4),G_G(:,2))/norm(p(:,3)-p(:,4))/...
    norm(G_G(:,2))<h2_h2
    A_A(1).L(1).p=p(:,1:3);
    A_A(1).L(2).p=p(:,3:5);
    NEWCURVE=NEWCURVE+1;
else
    if fun_value(p(:,4))>fun_value(p(:,3))&&fun_value(p(:,3))>...
        fun_value(p(:,2))
    else
        A_A(1).L(1).p=p(:,1:4);
        A_A(1).L(2).p=p(:,4:5);
        NEWCURVE=NEWCURVE+1;
    end
end

```

```

        end
    end
else
    if fun_value(p(:,3))>fun_value(p(:,2))&&fun_value(p(:,3))>...
        fun_value(p(:,4))
    else
        A_A(1).L(1).p=p(:,1:3);
        A_A(1).L(2).p=p(:,3:5);
        NEWCURVE=NEWCURVE+1;
    end
end
elseif size(p,2)==6
    G_G=ones(size(p,1),6);
    for ii_ii=1:4
        G_G(:,ii_ii)=-grad(p(:,ii_ii+1));
    end
    if dot(p(:,1)-p(:,2),G_G(:,1))/norm(p(:,1)-p(:,2))/...
        norm(G_G(:,1))<h2_h2&&dot(p(:,6)-p(:,5),G_G(:,4))/...
        norm(p(:,6)-p(:,5))/norm(G_G(:,4))<h2_h2
        A_A(1).L(1).p=p(:,1:3);
        A_A(1).L(2).p=p(:,3:6);
        NEWCURVE=NEWCURVE+1;
    elseif dot(p(:,1)-p(:,2),G_G(:,1))/norm(p(:,1)-p(:,2))/...
        norm(G_G(:,1))<h2_h2&&dot(p(:,6)-p(:,5),G_G(:,4))/...
        norm(p(:,6)-p(:,5))/norm(G_G(:,4))>h2_h2
    if dot(p(:,3)-p(:,2),G_G(:,2))/norm(p(:,3)-p(:,2))/...
        norm(G_G(:,2))<h2_h2||dot(p(:,5)-p(:,4),G_G(:,3))/...
        norm(p(:,5)-p(:,4))/norm(G_G(:,3))<h2_h2
        A_A(1).L(1).p=p(:,1:3);
        A_A(1).L(2).p=p(:,3:6);
        NEWCURVE=NEWCURVE+1;
    else
        if fun_value(p(:,2))>fun_value(p(:,3))&&fun_value(p(:,3))>...
            fun_value(p(:,4))&&fun_value(p(:,4))>fun_value(p(:,5))
        else
            A_A(1).L(1).p=p(:,1:2);
            A_A(1).L(2).p=p(:,2:6);

```

```

        NEWCURVE=NEWCURVE+1;
    end
end
elseif dot(p(:,1)-p(:,2),G_G(:,1))/norm(p(:,1)-p(:,2))/...
    norm(G_G(:,1))>h2_h2&&dot(p(:,6)-p(:,5),G_G(:,4))/...
    norm(p(:,6)-p(:,5))/norm(G_G(:,4))<h2_h2
    if dot(p(:,3)-p(:,2),G_G(:,2))/norm(p(:,3)-p(:,2))/...
        norm(G_G(:,2))>h2_h2||dot(p(:,5)-p(:,4),G_G(:,3))/...
        norm(p(:,5)-p(:,4))/norm(G_G(:,3))>h2_h2
        A_A(1).L(1).p=p(:,1:4);
        A_A(1).L(2).p=p(:,4:6);
        NEWCURVE=NEWCURVE+1;
    else
        if fun_value(p(:,5))>fun_value(p(:,4))&&fun_value(p(:,4))>...
            fun_value(p(:,3))&&fun_value(p(:,3))>fun_value(p(:,2))
        else
            A_A(1).L(1).p=p(:,1:5);
            A_A(1).L(2).p=p(:,5:6);
            NEWCURVE=NEWCURVE+1;
        end
    end
end
else
    if dot(p(:,3)-p(:,2),G_G(:,2))/norm(p(:,3)-p(:,2))/...
        norm(G_G(:,2))<h2_h2&&dot(p(:,5)-p(:,4),G_G(:,3))/...
        norm(p(:,5)-p(:,4))/norm(G_G(:,3))>h2_h2
    if fun_value(p(:,3))>fun_value(p(:,2))&&fun_value(p(:,4))>...
        fun_value(p(:,5))
    else
        if fun_value(p(:,3))<fun_value(p(:,2))
            A_A(1).L(1).p=p(:,1:3);
            A_A(1).L(2).p=p(:,3:6);
            NEWCURVE=NEWCURVE+1;
        else
            A_A(1).L(1).p=p(:,1:4);
            A_A(1).L(2).p=p(:,4:6);
            NEWCURVE=NEWCURVE+1;
        end
    end
end
end

```

```

end
elseif dot(p(:,3)-p(:,2),G_G(:,2))/norm(p(:,3)-p(:,2))/...
    norm(G_G(:,2))<h2_h2&&dot(p(:,5)-p(:,4),G_G(:,3))/...
    norm(p(:,5)-p(:,4))/norm(G_G(:,3))<h2_h2
if fun_value(p(:,4))>fun_value(p(:,3))&&fun_value(p(:,3))>...
    fun_value(p(:,2))
else
    A_A(1).L(1).p=p(:,1:4);
    A_A(1).L(2).p=p(:,4:6);
    NEWCURVE=NEWCURVE+1;
end
elseif dot(p(:,3)-p(:,2),G_G(:,2))/norm(p(:,3)-p(:,2))/...
    norm(G_G(:,2))>h2_h2&&dot(p(:,5)-p(:,4),G_G(:,3))/...
    norm(p(:,5)-p(:,4))/norm(G_G(:,3))>h2_h2
if fun_value(p(:,3))>fun_value(p(:,4))&&fun_value(p(:,4))>...
    fun_value(p(:,5))
else
    A_A(1).L(1).p=p(:,1:3);
    A_A(1).L(2).p=p(:,3:6);
    NEWCURVE=NEWCURVE+1;
end
else
    A_A(1).L(1).p=p(:,1:3);
    A_A(1).L(2).p=p(:,3:6);
    NEWCURVE=NEWCURVE+1;
end
end
end

if NEWCURVE==0
    A_A(1).L(1).p=p;
end
H=A_A(1).L;
end

```

*%this function seaches the saddle point on a curve. First, determine the
%control point with maximum energy. Then let this points climb up in order*

*%to coverage to the saddle point. And all other intermediate control points
%are minimized along the conjugate directions with positive eigenvalues.*

```
function H = saddle_search( p,c3_climb,c4,n,N_climb)
N_saddle=40;%maxi # iteration
A_A(1).p=p;
B_B(1).p=p;
i_i=1;
h_=0.1;%stop criteria based on the magnitude of the gradient
h_h=0.000001;%stop criteria based on change of function value
N_max=NMAX(p);%Control point with maxi energy on the curve
if N_max==1 || N_max==size(p,2)
    N_max=2;
end
a_a_for=1;
a_a_back=1;
a_a_tot=1;
m=size(p,2);
c_deg=1/(3*m);
th_deg=c_deg*norm(p(:,end)-p(:,1));
j_j=1;
while i_i<N_saddle
    i_i=i_i+1;
    A_A(i_i).p=ones(size(A_A(i_i-1).p));
    A_A(i_i).p(:,1)=A_A(i_i-1).p(:,1);
    A_A(i_i).p(:,end)=A_A(i_i-1).p(:,end);
    b_b=size(A_A(i_i).p,2);
    if N_max==b_b
        N_max=N_max+NMAX(A_A(i_i-1).p(:,N_max-2:N_max))-3;
    elseif N_max==1
        N_max=N_max+NMAX(A_A(i_i-1).p(:,N_max:N_max+2))-1;
    else
        N_max=N_max+NMAX(A_A(i_i-1).p(:,N_max-1:N_max+1))-2;
    end

    if N_max<ceil(size(A_A(i_i-1).p,2)/2)
        for ii_ii=2:N_max-1
```

```

A_A(i_i).p(:,ii_ii)=inter_mini(A_A(i_i-1).p(:,ii_ii-1),...
    A_A(i_i-1).p(:,ii_ii),c3_climb,c4,n,N_climb);
end

A_A(i_i).p(:,N_max)=climb_saddle(A_A(i_i-1).p(:,N_max-1),...
    A_A(i_i-1).p(:,N_max),c3_climb,c4,n,N_climb);
for ii_ii=N_max+1:ceil(size(A_A(i_i-1).p,2)/2)
    A_A(i_i).p(:,ii_ii)=inter_mini(A_A(i_i-1).p(:,ii_ii-1),...
        A_A(i_i-1).p(:,ii_ii),c3_climb,c4,n,N_climb);

end
for ii_ii=(ceil(size(A_A(i_i-1).p,2)/2)+1):(size(A_A(i_i-1).p,2)-1)
    A_A(i_i).p(:,ii_ii)=inter_mini(A_A(i_i-1).p(:,ii_ii+1),...
        A_A(i_i-1).p(:,ii_ii),c3_climb,c4,n,N_climb);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Redistribute the control points using degree elevation and reduction scheme
B_B(i_i).p=A_A(i_i).p;
c_c=index_loop(A_A(i_i).p);
if size(c_c,2)>0
    if max(c_c)<ceil(m/2)
        if a_a_for==0
            C_C(j_j).p(:,1:ceil(b_b/2)-1)=Be_degreduction(A_A(i_i).p(:,1:ceil(b_b/2)),ceil(b_b/2),th_deg);
            A_A(i_i).p(:,1:ceil(b_b/2)-1)=C_C(j_j).p;
            A_A(i_i).p(:,ceil(b_b/2):b_b-1)=B_B(i_i).p(:,ceil(b_b/2)+1:end);
            A_A(i_i).p(:,end)=[];
            j_j=j_j+1;
            a_a_for=a_a_for+1;
        else
            C_C(j_j).p(:,1:ceil(b_b/2)+1)=Be_degelevation(A_A(i_i).p(:,1:ceil(b_b/2)),ceil(b_b/2),th_deg);
            A_A(i_i).p(:,1:ceil(b_b/2)+1)=C_C(j_j).p;
            A_A(i_i).p(:,ceil(b_b/2)+2:b_b+1)=B_B(i_i).p(:,ceil(b_b/2)+1:end);
            j_j=j_j+1;
            a_a_for=0;
        end
    elseif min(c_c)>ceil(m/2)
        if a_a_back==0

```

```

C_C(j_j).p(:,1:b_b-ceil(b_b/2))=Be_degredution(...
    A_A(i_i).p(:,ceil(b_b/2):end),b_b-ceil(b_b/2)+1,th_deg);
A_A(i_i).p(:,ceil(b_b/2):b_b-1)=C_C(j_j).p;
A_A(i_i).p(:,end)=[];
j_j=j_j+1;
a_a_back=a_a_back+1;
else
C_C(j_j).p(:,1:b_b-ceil(b_b/2)+2)=Be_degelevation(...
    A_A(i_i).p(:,ceil(b_b/2):end),b_b-ceil(b_b/2)+1,th_deg);
A_A(i_i).p(:,ceil(b_b/2):b_b+1)=C_C(j_j).p;
j_j=j_j+1;
a_a_back=0;
end
else
if a_a_tot==0
A_A(i_i).p(:,1:(b_b-1))=Be_degredution(A_A(i_i).p,b_b,th_deg);
A_A(i_i).p(:,end)=[];
a_a_tot=a_a_tot+1;
else
A_A(i_i).p(:,1:(b_b+1))=Be_degelevation(A_A(i_i).p,b_b,th_deg);
a_a_tot=0;
end
end
end
end
%%%%%%%%%%
elseif N_max>ceil(size(A_A(i_i-1).p,2)/2)
for ii_ii=2:ceil(size(A_A(i_i-1).p,2)/2)
A_A(i_i).p(:,ii_ii)=inter_mini(A_A(i_i-1).p(:,ii_ii-1),...
    A_A(i_i-1).p(:,ii_ii),c3_climb,c4,n,N_climb);
end
for ii_ii=(ceil(size(A_A(i_i-1).p,2)/2)+1):N_max-1
A_A(i_i).p(:,ii_ii)=inter_mini(A_A(i_i-1).p(:,ii_ii-1),...
    A_A(i_i-1).p(:,ii_ii),c3_climb,c4,n,N_climb);
end
A_A(i_i).p(:,N_max)=climb_saddle(A_A(i_i-1).p(:,N_max+1),...
    A_A(i_i-1).p(:,N_max),c3_climb,c4,n,N_climb);
for ii_ii=N_max+1:(size(A_A(i_i-1).p,2)-1)

```



```

A_A(i_i).p(:,ii_ii)=inter_mini(A_A(i_i-1).p(:,ii_ii+1),...
    A_A(i_i-1).p(:,ii_ii),c3_climb,c4,n,N_climb);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       redistribute the control points using degree elevation and reduction scheme
B_B(i_i).p=A_A(i_i).p;
c_c=index_loop(A_A(i_i).p);
if size(c_c,2)>0
    if max(c_c)<ceil(m/2)
        if a_a_for==0
            C_C(j_j).p(:,1:ceil(b_b/2)-1)=Be_degreduction(A_A(i_i).p(:,1:ceil(b_b/2)),ceil(b_b/2),th_deg);
            A_A(i_i).p(:,1:ceil(b_b/2)-1)=C_C(j_j).p;
            A_A(i_i).p(:,ceil(b_b/2):b_b-1)=B_B(i_i).p(:,ceil(b_b/2)+1:end);
            A_A(i_i).p(:,end)=[];
            j_j=j_j+1;
            a_a_for=a_a_for+1;
        else
            C_C(j_j).p(:,1:ceil(b_b/2)+1)=Be_degelevation(A_A(i_i).p(:,1:ceil(b_b/2)),ceil(b_b/2),th_deg);
            A_A(i_i).p(:,1:ceil(b_b/2)+1)=C_C(j_j).p;
            A_A(i_i).p(:,ceil(b_b/2)+2:b_b+1)=B_B(i_i).p(:,ceil(b_b/2)+1:end);
            j_j=j_j+1;
            a_a_for=0;
        end
    elseif min(c_c)>ceil(m/2)
        if a_a_back==0
            C_C(j_j).p(:,1:b_b-ceil(b_b/2))=Be_degreduction(...
                A_A(i_i).p(:,ceil(b_b/2):end),b_b-ceil(b_b/2)+1,th_deg);
            A_A(i_i).p(:,ceil(b_b/2):b_b-1)=C_C(j_j).p;
            A_A(i_i).p(:,end)=[];
            j_j=j_j+1;
            a_a_back=a_a_back+1;
        else
            C_C(j_j).p(:,1:b_b-ceil(b_b/2)+2)=Be_degelevation(...
                A_A(i_i).p(:,ceil(b_b/2):end),b_b-ceil(b_b/2)+1,th_deg);
            A_A(i_i).p(:,ceil(b_b/2):b_b+1)=C_C(j_j).p;
            j_j=j_j+1;
            a_a_back=0;
        end
    end
end

```

```

end
else
if a_a_tot==0
A_A(i_i).p(:,1:(b_b-1))=Be_degreduction(A_A(i_i).p,b_b,th_deg);
A_A(i_i).p(:,end)=[];
a_a_tot=a_a_tot+1;
else
A_A(i_i).p(:,1:(b_b+1))=Be_degelevation(A_A(i_i).p,b_b,th_deg);
a_a_tot=0;
end
end
end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

else
for ii_ii=2:ceil(size(A_A(i_i-1).p,2)/2)-1
A_A(i_i).p(:,ii_ii)=inter_mini(A_A(i_i-1).p(:,ii_ii-1),...
A_A(i_i-1).p(:,ii_ii),c3_climb,c4,n,N_climb);
end
A_A(i_i).p(:,N_max)=climb_saddle(A_A(i_i-1).p(:,N_max-1),...
A_A(i_i-1).p(:,N_max),c3_climb,c4,n,N_climb);
for ii_ii=(ceil(size(A_A(i_i-1).p,2)/2)+1):(size(A_A(i_i-1).p,2)-1)
A_A(i_i).p(:,ii_ii)=inter_mini(A_A(i_i-1).p(:,ii_ii+1),...
A_A(i_i-1).p(:,ii_ii),c3_climb,c4,n,N_climb);
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% redistribute the control points using degree elevation and reduction scheme

```

```

B_B(i_i).p=A_A(i_i).p;
c_c=index_loop(A_A(i_i).p);
if size(c_c,2)>0
if max(c_c)<ceil(m/2)
if a_a_for==0
C_C(j_j).p(:,1:ceil(b_b/2)-1)=Be_degreduction(A_A(i_i).p(:,1:ceil(b_b/2)),ceil(b_b/2),th_deg);
A_A(i_i).p(:,1:ceil(b_b/2)-1)=C_C(j_j).p;
A_A(i_i).p(:,ceil(b_b/2):b_b-1)=B_B(i_i).p(:,ceil(b_b/2)+1:end);
A_A(i_i).p(:,end)=[];
j_j=j_j+1;

```

```

    a_a_for=a_a_for+1;
else
    C_C(j_j).p(:,1:ceil(b_b/2)+1)=Be_degelevation(A_A(i_i).p(:,1:ceil(b_b/2)),ceil(b_b/2),th_deg);
    A_A(i_i).p(:,1:ceil(b_b/2)+1)=C_C(j_j).p;
    A_A(i_i).p(:,ceil(b_b/2)+2:b_b+1)=B_B(i_i).p(:,ceil(b_b/2)+1:end);
    j_j=j_j+1;
    a_a_for=0;
end
elseif min(c_c)>ceil(m/2)
if a_a_back==0
    C_C(j_j).p(:,1:b_b-ceil(b_b/2))=Be_degredution(...
        A_A(i_i).p(:,ceil(b_b/2):end),b_b-ceil(b_b/2)+1,th_deg);
    A_A(i_i).p(:,ceil(b_b/2):b_b-1)=C_C(j_j).p;
    A_A(i_i).p(:,end)=[];
    j_j=j_j+1;
    a_a_back=a_a_back+1;
else
    C_C(j_j).p(:,1:b_b-ceil(b_b/2)+2)=Be_degelevation(...
        A_A(i_i).p(:,ceil(b_b/2):end),b_b-ceil(b_b/2)+1,th_deg);
    A_A(i_i).p(:,ceil(b_b/2):b_b+1)=C_C(j_j).p;
    j_j=j_j+1;
    a_a_back=0;
end
else
if a_a_tot==0
    A_A(i_i).p(:,1:(b_b-1))=Be_degredution(A_A(i_i).p,b_b,th_deg);
    A_A(i_i).p(:,end)=[];
    a_a_tot=a_a_tot+1;
else
    A_A(i_i).p(:,1:(b_b+1))=Be_degelevation(A_A(i_i).p,b_b,th_deg);
    a_a_tot=0;
end
end
end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

end

```

```

if i_i>2
    if abs((fun_value(B_B(i_i).p(:,N_max))-fun_value...
        (B_B(i_i-2).p(:,N_max)))/fun_value(B_B(i_i-2).p(:,N_max)))<h_h &&...
        norm(grad(B_B(i_i).p(:,N_max)))<h_
        break
    end
end
end
H=B_B(i_i).p;
end

```

*%this function determines the index of the point with maxi function value
%in p*

```

function H = NMAX( p )
for ii_ii=1:size(p,2)
    FV(ii_ii)=fun_value(p(:,ii_ii));
end
MAX=max(FV);
H=find(FV==MAX);
end

```

%this function determines the maxi function value point in p

```

function H = NMAX_VALUE( p )
for ii_ii=1:size(p,2)
    FV(ii_ii)=fun_value(p(:,ii_ii));
end
MAX=max(FV);
a_a=find(FV==MAX);
H=p(:,a_a);
end

```

%this function maximize p2 in one conjugate direction with negative

%eigenvalues and then minimize p2 in all other conjugate directions.

```
function H = climb_saddle( p1,p2,c3,c4,n,N_climb )
% N_climb: #of mini-step in conjugate direction
b1=0.001;%threshold for convergence criteria
b2=0.000001;%threshold for convergence criteria
h_h1=-0.001;
x1=ones(n,n);%x1: matrix to store the positions of p2 interatively
%g1 is a n by(n-1)matrix to store the gradient g1,g2...in Beal's formula
g1=ones(n,n-1);
s=ones(n,n);
s(:,1)=p2-p1; %direction with negative eigenvalue(maximization direction)
%Locate the maximum point along s0=s(:,1) in order to determine g1
%in Beal's formula
x1(:,1)=p2;
xmid=1/2*(p1+p2);
g0(:,1)=grad(xmid);
ii=1;
X1(:,1)=p2;
G1(:,1)=grad(p2);
c3_c3=c3;
while ii<N_climb&&abs(dot(G1(:,ii),s(:,1))/dot(s(:,1),s(:,1)))>b2
    ii=ii+1;
    X1(:,ii)=X1(:,ii-1)+c3_c3*dot(G1(:,ii-1),s(:,1))/norm(s(:,1))^2*s(:,1);
    G1(:,ii)=grad(X1(:,ii));
    if dot(G1(:,ii-1),s(:,1))*dot(G1(:,ii),s(:,1))/norm(s(:,1))^2<h_h1
        c3_c3=c4*c3_c3;
    end
end
x1(:,1)=X1(:,ii); %maximum point along the s0
g1(:,1)=grad(x1(:,1));
%calculate s(:,2)represents s1 in Beal's formula (first conjugate direction
%with positive eigenvalue)
if abs(dot(s(:,1),(g1(:,1)-g0(:,1))))>b1
    s(:,2)=-g1(:,1)+dot(g1(:,1),(g1(:,1)-g0(:,1)))/...
        dot(s(:,1),(g1(:,1)-g0(:,1)))*s(:,1);
else
```

```

    s(:,2)=-g1(:,1);
end

%Line minimization in direction s(:,2) for the maximum point along s(:,1)
ii=1;
X1(:,1)=x1(:,1);
G1(:,1)=g1(:,1);
c3_c3=c3;

while ii<N_climb&&abs(dot(G1(:,ii),s(:,2))/dot(s(:,2),s(:,2)))>b2
    ii=ii+1;
    X1(:,ii)=X1(:,ii-1)+c3_c3*dot(-G1(:,ii-1),s(:,2))/norm(s(:,2))^2*s(:,2);
    G1(:,ii)=grad(X1(:,ii));
    if dot(G1(:,ii-1),s(:,1))*dot(G1(:,ii),s(:,1))/norm(s(:,1))^2<h_h1
        c3_c3=c4*c3_c3;
    end
end

x1(:,2)=X1(:,ii);
g1(:,2)=grad(x1(:,2));

%if the dimension of the PES is larger than two, then minimize
%p2 along all the conjugate directions with negative eigenvalues
%iteratively.
jj=3;
while jj<=n

    if abs(dot(s(:,1),(g1(:,1)-g0(:,1))))>b1
        s(:,jj)=-g1(:,jj-1)+dot(g1(:,jj-1),(g1(:,1)-g0(:,1)))/...
            dot(s(:,1),(g1(:,1)-g0(:,1)))*s(:,1)+dot(g1(:,jj-1),...
            g1(:,jj-1))/dot(g1(:,jj-2),g1(:,jj-2))*s(:,jj-1);
    else
        s(:,jj)=-g1(:,jj-1)+dot(g1(:,jj-1),g1(:,jj-1))/...
            dot(g1(:,jj-2),g1(:,jj-2))*s(:,jj-1);
    end

%minimize p2 along s(:,jj) direction

```

```

ii=1;
X1(:,1)=x1(:,jj-1);
G1(:,1)=g1(:,jj-1);
c3_c3=c3;
while ii<N_climb&&abs(dot(G1(:,ii),s(:,jj))/dot(s(:,jj),s(:,jj)))>b2
    ii=ii+1;
    X1(:,ii)=X1(:,ii-1)+c3_c3*dot(-G1(:,ii-1),s(:,jj))/...
        norm(s(:,jj))^2*s(:,jj);
    G1(:,ii)=grad(X1(:,ii));
    if dot(G1(:,ii-1),s(:,1))*dot(G1(:,ii),s(:,1))/norm(s(:,1))^2<h_h1
        c3_c3=c4*c3_c3;
    end
end
x1(:,jj)=X1(:,ii);
g1(:,jj)=grad(x1(:,jj));
jj=jj+1;
end

H=x1(:,jj-1);
end

```

REFERENCES

- [1] J.M. Yeomans, Statistical mechanics of phase transitions, Oxford University Press, New York, 2002.
- [2] P. Papon, J. Leblond, P.H.E. Meijer, The physics of phase transitions: concepts and applications, Springer-Verlag, Berlin, 2006.
- [3] K.J. Laidler, M.C. King, Development of transition-state theory, The Journal of Physical Chemistry, 87 (1983) 2657-2664.
- [4] A.F. Voter, Radiation Effects in Solids, Springer, NATO Publishing Unit, Dordrecht, The Netherlands, 2005.
- [5] H. Jonsson, G. Mills, K. Jacobsen, Classical and Quantum Dynamics in Condensed Phase Simulations, in, World Scientific, Hackensack, NJ, 1998, pp. 385-404.
- [6] E. W., W. Ren, E. Vanden-Eijnden, String method for the study of rare events, Physical Review B, 66 (2002) 052301(052301-052304).

- [7] G. Henkelman, H. Jónsson, A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives, *The Journal of chemical physics*, 111 (1999) 7010-7022.
- [8] H. Eyring, M. Polanyi, Uber einfache Gasreaktionen, *Zeitschrift für physikalische Chemie B*, 12 (1931) 279.
- [9] G.H. Vineyard, Frequency factors and isotope effects in solid state rate processes, *Journal of Physics and Chemistry of Solids*, 3 (1957) 121-127.
- [10] D.G. Truhlar, B.C. Garrett, Variational transition-state theory, *Accounts of Chemical Research*, 13 (1980) 440-448.
- [11] W.H. Miller, N.C. Handy, J.E. Adams, Reaction path Hamiltonian for polyatomic molecules, *The Journal of chemical physics*, 72 (1980) 99.
- [12] L. Chen, S. Ying, T. Ala-Nissila, Finding transition paths and rate coefficients through accelerated Langevin dynamics, *Physical Review E*, 65 (2002) 042101.
- [13] G.A. Mansoori, Principles of nanotechnology: molecular-based study of condensed matter in small systems, World Scientific Pub Co Inc, MA, USA, 2005.
- [14] D.W. Brenner, C. Bean, Empirical potential for hydrocarbons for use in simulating the chemical vapor deposition of diamond films, *Physical Review B*, 42 (1990) 9458-9471.
- [15] M.G. Burke, S.N. Yaliraki, Exploring model energy and geometry surfaces using sum of squares decompositions, *Journal of Chemical Theory and Computation*, 2 (2006) 575-587.
- [16] T. Hollebeek, T.S. Ho, H. Rabitz, Constructing multidimensional molecular potential energy surfaces from ab initio data, *Annual review of physical chemistry*, 50 (1999) 537-570.
- [17] S. Sato, On a new method of drawing the potential energy surface, *The Journal of chemical physics*, 23 (1955) 592-593.
- [18] K.C. Thompson, M.J.T. Jordan, M.A. Collins, Molecular potential energy surfaces by interpolation in Cartesian coordinates, *The Journal of chemical physics*, 108 (1998) 564-578.
- [19] D.G. Truhlar, R. Steckler, M.S. Gordon, Potential energy surfaces for polyatomic reaction dynamics, *Chemical Reviews*, 87 (1987) 217-236.
- [20] R.J. Duchovic, Y.L. Volobuev, G.C. Lynch, D.G. Truhlar, T.C. Allison, A.F. Wagner, B.C. Garrett, J.C. Corchado, POTLIB 2001: A potential energy surface library for chemical systems, *Computer physics communications*, 144 (2002) 169-187.
- [21] D. Alhat, V. Lasrado, Y. Wang, A Review of Recent Phase Transition Simulation Methods: Saddle Point Search, in, ASME, 2008.
- [22] G. Henkelman, G. Johannesson, H. Jónsson, Theoretical methods in condensed phase chemistry, *Progress in theoretical chemistry and physics*, 5 (2000).
- [23] V. Lasrado, D. Alhat, Y. Wang, A Review of Recent Phase Transition Simulation Methods: Transition Path Search, in, ASME, 2008.
- [24] R. Olsen, G. Kroes, G. Henkelman, A. Arnaldsson, H. Jónsson, Comparison of methods for finding saddle points without knowledge of the final states, *The Journal of chemical physics*, 121 (2004) 9776.

- [25] H.B. Schlegel, Exploring potential energy surfaces for chemical reactions: an overview of some practical methods, *Journal of Computational Chemistry*, 24 (2003) 1514-1527.
- [26] N. Mousseau, G. Barkema, Traveling through potential energy landscapes of disordered materials: The activation-relaxation technique, *Physical Review E*, 57 (1998) 2419.
- [27] G. Henkelman, H. Jónsson, A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives, *The Journal of chemical physics*, 111 (1999) 7010.
- [28] M.J.S. Dewar, E.F. Healy, J.J.P. Stewart, Location of transition states in reaction mechanisms, *J. Chem. Soc., Faraday Trans. 2*, 80 (1984) 227-233.
- [29] I.V. Ionova, E.A. Carter, Ridge method for finding saddle points on potential energy surfaces, *The Journal of chemical physics*, 98 (1993) 6377.
- [30] C.J. Cerjana, W.H. Miller, On finding transition states, *The Journal of chemical physics*, 75 (1981) 2800-2806.
- [31] K. Fukui, S. Kato, H. Fujimoto, Constituent analysis of the potential gradient along a reaction coordinate. Method and an application to methane+ tritium reaction, *Journal of the American Chemical Society*, 97 (1975) 1-7.
- [32] K. Müller, Reaction paths on multidimensional energy hypersurfaces, *Angewandte Chemie International Edition in English*, 19 (1980) 1-13.
- [33] C. Choi, R. Elber, Reaction path study of helix formation in tetrapeptides: Effect of side chains, *The Journal of chemical physics*, 94 (1991) 751.
- [34] R. Czerminski, R. Elber, Self - avoiding walk between two fixed points as a tool to calculate reaction paths in large molecular systems, *International Journal of Quantum Chemistry*, 38 (1990) 167-185.
- [35] R. Elber, M. Karplus, A method for determining reaction paths in large molecules: application to myoglobin, *Chemical physics letters*, 139 (1987) 375-380.
- [36] R.E. Gillilan, K.R. Wilson, Shadowing, rare events, and rubber bands. A variational Verlet algorithm for molecular dynamics, *The Journal of chemical physics*, 97 (1992) 1757.
- [37] L.R. Pratt, A statistical method for identifying transition states in high dimensional problems, *The Journal of chemical physics*, 85 (1986) 5045.
- [38] E. Sevick, A. Bell, D. Theodorou, A chain of states method for investigating infrequent event processes occurring in multistate, multidimensional systems, *The Journal of chemical physics*, 98 (1993) 3196.
- [39] A. Ulitsky, R. Elber, A new technique to calculate steepest descent paths in flexible polyatomic systems, *The Journal of chemical physics*, 92 (1990) 1510-1511.
- [40] G. Henkelman, H. Jónsson, Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points, *The Journal of chemical physics*, 113(22) (2000) 9978-9985.
- [41] S.A. Trygubenko, D.J. Wales, A doubly nudged elastic band method for finding transition states, *The Journal of chemical physics*, 120(5) (2004) 2082-2094.

- [42] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, *Mathematical programming*, 45 (1989) 503-528.
- [43] G. Henkelman, B.P. Uberuaga, H. Jónsson, A climbing image nudged elastic band method for finding saddle points and minimum energy paths, *The Journal of chemical physics*, 113(22) (2000) 9901-9904.
- [44] T. Zhu, J. Li, A. Samanta, H.G. Kim, S. Suresh, Interfacial plasticity governs strain rate sensitivity and ductility in nanostructured metals, *Proceedings of the National Academy of Sciences*, 104 (2007) 3031-3036.
- [45] I.F. Galvan, M.J. Field, Improving the efficiency of the NEB reaction path finding algorithm, *Journal of Computational Chemistry*, 29 (2008) 139-143.
- [46] Y. Kumeda, D.J. Wales, L.J. Munro, Transition states and rearrangement mechanisms from hybrid eigenvector-following and density functional theory.: Application to C10H10 and defect migration in crystalline silicon, *Chemical physics letters*, 341 (2001) 185-194.
- [47] L.J. Munro, D.J. Wales, Defect migration in crystalline silicon, *Physical Review B*, 59 (1999) 3969-3980.
- [48] W. Ren, Higher order string method for finding minimum energy paths, *Communications in Mathematical Sciences*, 1 (2003) 377-384.
- [49] W. E, W. Ren, E. Vanden-Eijnden, Simplified and improved string method for computing the minimum energy paths in barrier-crossing events, *Journal of Chemical Physics*, 126 (2007) 164103(164101) -164103(164108).
- [50] B. Peters, A. Heyden, A.T. Bell, A. Chakraborty, A growing string method for determining transition states: Comparison to the nudged elastic band and string methods, *The Journal of chemical physics*, 120 (2004) 7877-7886.
- [51] S.K. Burger, W. Yang, Quadratic string method for determining the minimum-energy path based on multiobjective optimization, *The Journal of chemical physics*, 124 (2006) 054109(054101)- 054109(054112).
- [52] S. Fischer, M. Karplus, Conjugate peak refinement: an algorithm for finding reaction paths and accurate transition states in systems with many degrees of freedom, *Chemical physics letters*, 194 (1992) 252-261.
- [53] L. Chen, S. Ying, T. Ala-Nissila, Finding transition paths and rate coefficients through accelerated Langevin dynamics, *Physical Review E*, 65 (2002) 042101(042101-042104).
- [54] D. Passerone, M. Ceccarelli, M. Parrinello, A concerted variational strategy for investigating rare events, *The Journal of chemical physics*, 118 (2003) 2025.
- [55] B.K. Dey, P.W. Ayers, A Hamilton–Jacobi type equation for computing minimum potential energy paths, *Molecular Physics*, 104 (2006) 541-558.
- [56] E.M.L. Beale, *Numerical methods for non-linear optimization* Academic Press, London, 1972.
- [57] J. Sinclair, R. Fletcher, A new method of saddle-point location for the calculation of defect migration energies, *Journal of Physics C: Solid State Physics*, 7 (1974) 864.

- [58] Y. Saad, Y. Saad, Iterative methods for sparse linear systems, PWS publishing company Boston, 1996.
- [59] J. Simons, P. Joergensen, H. Taylor, J. Ozment, Walking on Potential Energy Surfaces, The Journal of Physical Chemistry, 87 (1983) 2745-2753.
- [60] A. Banerjee, N. Adams, J. Simons, R. Shepard, Search for Stationary Points on Surfaces, The Journal of Physical Chemistry, 89 (1985) 52-57.
- [61] A. Heyden, A.T. Bell, F.J. Keil, Efficient methods for finding transition states in chemical reactions: Comparison of improved dimer method and partitioned rational function optimization method, The Journal of chemical physics, 123 (2005) 224101-224114.
- [62] W. Quapp, M. Hirsch, O. Imig, D. Heidrich, Searching for saddle points of potential energy surfaces by following a reduced gradient, Journal of Computational Chemistry, 19 (1998) 1087-1100.
- [63] J.M. Anglada, E. Besalú, J.M. Bofill, R. Crehuet, On the quadratic reaction path evaluated in a reduced potential energy surface model and the problem to locate transition states*, Journal of Computational Chemistry, 22 (2001) 387-406.
- [64] M. Hirsch, W. Quapp, Improved RGF method to find saddle points, Journal of computational chemistry, 23 (2002) 887-894.
- [65] T.A. Halgren, W.N. Lipscomb, The synchronous-transit method for determining reaction pathways and locating molecular transition states, Chemical Physics Letters, 49 (1977) 225-232.
- [66] N. Govind, M. Petersen, G. Fitzgerald, D. King-Smith, J. Andzelm, A generalized synchronous transit method for transition state location, Computational materials science, 28 (2003) 250-258.
- [67] R.A. Miron, K.A. Fichthorn, The Step and Slide method for finding saddle points on multidimensional potential surfaces, The Journal of chemical physics, 115 (2001) 8742.
- [68] Y. Lin, M.A. Stadtherr, Locating stationary points of sorbate-zeolite potential energy surfaces using interval analysis, The Journal of chemical physics, 121 (2004) 10159.
- [69] M.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, in, NBS, 1952.
- [70] J.R. Shewchuk, An introduction to the conjugate gradient method without the agonizing pain, in, Carnegie Mellon University, Pittsburgh, PA, 1994.
- [71] G.H. Golub, D.P. O'Leary, Some history of the conjugate gradient and Lanczos algorithms: 1948-1976, SIAM review, 31 (1989) 50-102.
- [72] R. Fletcher, C. Reeves, Function minimization by conjugate gradients, The computer journal, 7 (1964) 149-154.
- [73] E. POLA, G. Ribiere, Note sur la convergence de methodes de directions conjuguées, Rev Française Informat Recherche Operationelle, 3e Anné, 16 (1969) 35-43.
- [74] G. Zoutendijk, Nonlinear programming, computational methods, Integer and nonlinear programming, 143 (1970) 37-86.

- [75] M. Al-Baali, Descent property and global convergence of the Fletcher—Reeves method with inexact line search, *IMA Journal of Numerical Analysis*, 5 (1985) 121-124.
- [76] M. Powell, Nonconvex minimization calculations and the conjugate gradient method, *Numerical Analysis*, (1984) 122-141.
- [77] Y. Hu, C. Storey, Global convergence result for conjugate gradient methods, *Journal of Optimization Theory and Applications*, 71 (1991) 399-405.
- [78] J.C. Gilbert, J. Nocedal, Global convergence properties of conjugate gradient methods for optimization, *SIAM Journal on Optimization*, 2 (1992) 21-42.
- [79] L. Guanghui, H. Jiye, Y. Hongxia, Global convergence of the Fletcher-Reeves algorithm with inexact linesearch, *Applied Mathematics-A Journal of Chinese Universities*, 10 (1995) 75-82.
- [80] Y. Dai, J. Han, G. Liu, D. Sun, H. Yin, Y.X. Yuan, Convergence properties of nonlinear conjugate gradient methods, *SIAM Journal on Optimization*, 10 (2000) 345-358.
- [81] M.E. Mortenson, *Geometric modeling*, John Wiley, New York, 1985.
- [82] M.A. Watkins, A.J. Worsey, Degree reduction of B ézier curves, *Computer-Aided Design*, 20 (1988) 398-405.
- [83] M. Eck, Degree reduction of B ézier curves, *Computer Aided Geometric Design*, 10 (1993) 237-251.
- [84] A.R. Forrest, Interactive interpolation and approximation by B ézier polynomials, *The Computer Journal*, 15 (1972) 71-79.
- [85] M. Eck, Least squares degree reduction of B ézier curves, *Computer-Aided Design*, 27 (1995) 845-851.
- [86] P. Bogacki, S.E. Weinstein, Y. Xu, Degree reduction of B ézier curves by uniform approximation with endpoint interpolation, *Computer-Aided Design*, 27 (1995) 651-661.
- [87] G. Brunnett, T. Schreiber, J. Braun, The geometry of optimal degree reduction of B ézier curves, *Computer Aided Geometric Design*, 13 (1996) 773-788.
- [88] H. Kim, S. Moon, Degree reduction of B ézier curves by L^1 -Approximation with endpoint interpolation, *Computers & Mathematics with Applications*, 33 (1997) 67-77.
- [89] H. Kim, Y. Ahn, Good degree reduction of B ézier curves using Jacobi polynomials, *Computers & Mathematics with Applications*, 40 (2000) 1205-1215.
- [90] Y.J. Ahn, Using Jacobi polynomials for degree reduction of B ézier curves with C^k -constraints, *Computer Aided Geometric Design*, 20 (2003) 423-434.
- [91] G.D. Chen, G.J. Wang, Optimal multi-degree reduction of B ézier curves with constraints of endpoints continuity, *Computer Aided Geometric Design*, 19 (2002) 365-377.
- [92] H. Sunwoo, Matrix representation for multi-degree reduction of B ézier curves, *Computer Aided Geometric Design*, 22 (2005) 261-273.

- [93] L. Lu, G. Wang, Application of Chebyshev II–Bernstein basis transformations to degree reduction of Bézier curves, *Journal of Computational and Applied Mathematics*, 221 (2008) 52-65.
- [94] L. Lu, G. Wang, Optimal multi-degree reduction of Bézier curves with G2-continuity, *Computer Aided Geometric Design*, 23 (2006) 673-683.
- [95] A. Rababah, B.G. Lee, J. Yoo, A simple matrix form for degree reduction of Bézier curves using Chebyshev–Bernstein basis transformations, *Applied mathematics and computation*, 181 (2006) 310-318.
- [96] P. Woźny, S. Lewanowicz, Multi-degree reduction of Bézier curves with constraints, using dual Bernstein basis polynomials, *Computer Aided Geometric Design*, 26 (2009) 566-579.
- [97] X.D. Chen, W. Ma, J.C. Paul, Multi-degree reduction of Bézier curves using reparameterization, *Computer-Aided Design*, 43 (2011) 161-169.
- [98] G.E. Farin, *Curves and Surfaces for Computer-Aided Geometric Design*, Academic Press, Boston, 1993.
- [99] J. Polanyi, W. Wong, Location of energy barriers. I. Effect on the dynamics of reactions A+BC, *The Journal of Chemical Physics*, 51 (1969) 1439.