A New Algorithm for Finding Critical Points of Potential Energy Surfaces

Bal Krishan

A dissertation submitted for the partial fulfillment of
BS-MS dual degree in Science

INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH, MOHALI

November 2017

Certificate of Examination

This is to certify that the dissertation titled "**A New Algorithm for Finding Critical Points of Potential Energy Surfaces**" submitted by **Mr. Bal Krishan (Reg. No. MS12008)** for the partial fulfilment of BS-MS dual degree programme of the Institute, has been examined by the thesis committee duly appointed by the Institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

Dr. Abhishek Chaudhuri Dr. Shashi Bhushan Pandit Dr. P. Balanarayan
(Supervisor)

Declaration

The work presented in this dissertation has been carried out by me with Dr. P. Balanarayan at Indian Institute of Science Education and Research, Mohali. This work has not been submitted in part or in full for a degree, a diploma, or a fellowship to any other university or institute. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due acknowledgement of collaborative research and discussions. This thesis is a bonafide record of original work done by me and all sources listed within have been detailed in the bibliography.

Bal Krishan
(Candidate)
Dated: November 26, 2017

In my capacity as the supervisor of the candidate's project work, I certify that the above statements by the candidate are true to the best of my knowledge.

Dr. P. Balanarayan
(Supervisor)

Acknowledgements

I would like to express my sincere gratitude to Dr. P. Balanarayan, for his great confidence and support throughout this project. I also thank Dr. Abhishek Chaudhuri and Dr. Sashi Bhushan Pandit for reading drafts of this dissertation and providing valuable comments. I am grateful to Prashant for his constant support and many valuable discussions we had. I would like to acknowledge Abhijeet, Deep Raj and Prateek for various helpful discussions.

Dedicated to the process of unlearning.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In chemical physics, molecular potentials or potential energy hypersurfaces (PES) play an important role in understanding structure and dynamics of complex systems such as proteins[1]. These energy surfaces are in general complicated functions of several variables and are evaluated on a finite grid by ab-initio quantum mechanical methods or empirical force fields. Traditionally, these grid potentials are then fitted to obtain analytic expressions to evaluate functions and its derivatives. However, with continuing advances in software and computational power, nowadays it is possible to obtain potential and derivatives on the fly using ab-initio methods. Molecular potentials have a large number of critical points (maxima, minima and saddle points) which correspond to equilibrium geometries and transition states. Important information about reaction dynamics can also be extracted by determining classical trajectories on PES connecting these points. Because regions of PES away from stable structures are not accessible experimentally, theoretical study of PESs provide great insights about molecular structure and dynamics and a number of numerical methods have been developed to locate and analyze critical points on PES.

A critical point $\mathbf{x}_0 \equiv (x_1, x_2, .., x_n)$ of a function of several variables, $f(x_1, x_2, .., x_n)$ is a point where first order partial derivative with respect to each variable vanishes i.e. gradient $\vec{\nabla} f$ of the function vanishes at point $\mathbf{x}_0$.

$$\nabla f \equiv \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = 0 \tag{1.1}$$

A critical point corresponds to either a minimum or maximum or a saddle point on the PES. Nature of a critical point can be determined by evaluating the Hessian matrix $[\nabla^2 f]$, at that point and diagnolizing it to obtain its eigenvalues.

$$\nabla^2 f \equiv \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & & \vdots \\ \vdots & & \ddots & \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{1.2}$$

If none of the eigenvalues is zero, the critical point is called nondegenerate and is characterized by the rank (R) of the Hessian matrix which is equal to number of nonzero eigenvalues and algebraic sum of signs of eigenvalues known as signature (S). For a function of three variables, four different types of critical points are possible: a minimum (3,+3), a maximum (3,-3) and two kinds of saddle points (3,+1) and (3,-1) in the (R,S) representation[2].

1.1 Problem of Unconstrained Optimization

In mathematics, the problem of finding minima of a real function $f : \mathbb{R}^n \to \mathbb{R}$ with no restraints on its arguments $\mathbf{x} \in \mathbb{R}^n$ is called unconstrained optimization problem. Maximum of a function corresponds to minimum of negative of that function and can be found in the same way. Numerical procedures which find minima are called optimizers. In practice often a global minimum is required and is selected from the set of solutions however there also exist algorithms and heuristics[1] called global optimizers that tends to find a global minimum by avoiding unnecessary minima. Other methods which finds minima or maxima in a nearby region are called local optimizers. Unconstrained optimization problems broadly fall in two categories[3,4]:

[1] A heuristic is a procedure designed to find approximate solutions of a problem when exact methods are too slow or not available.

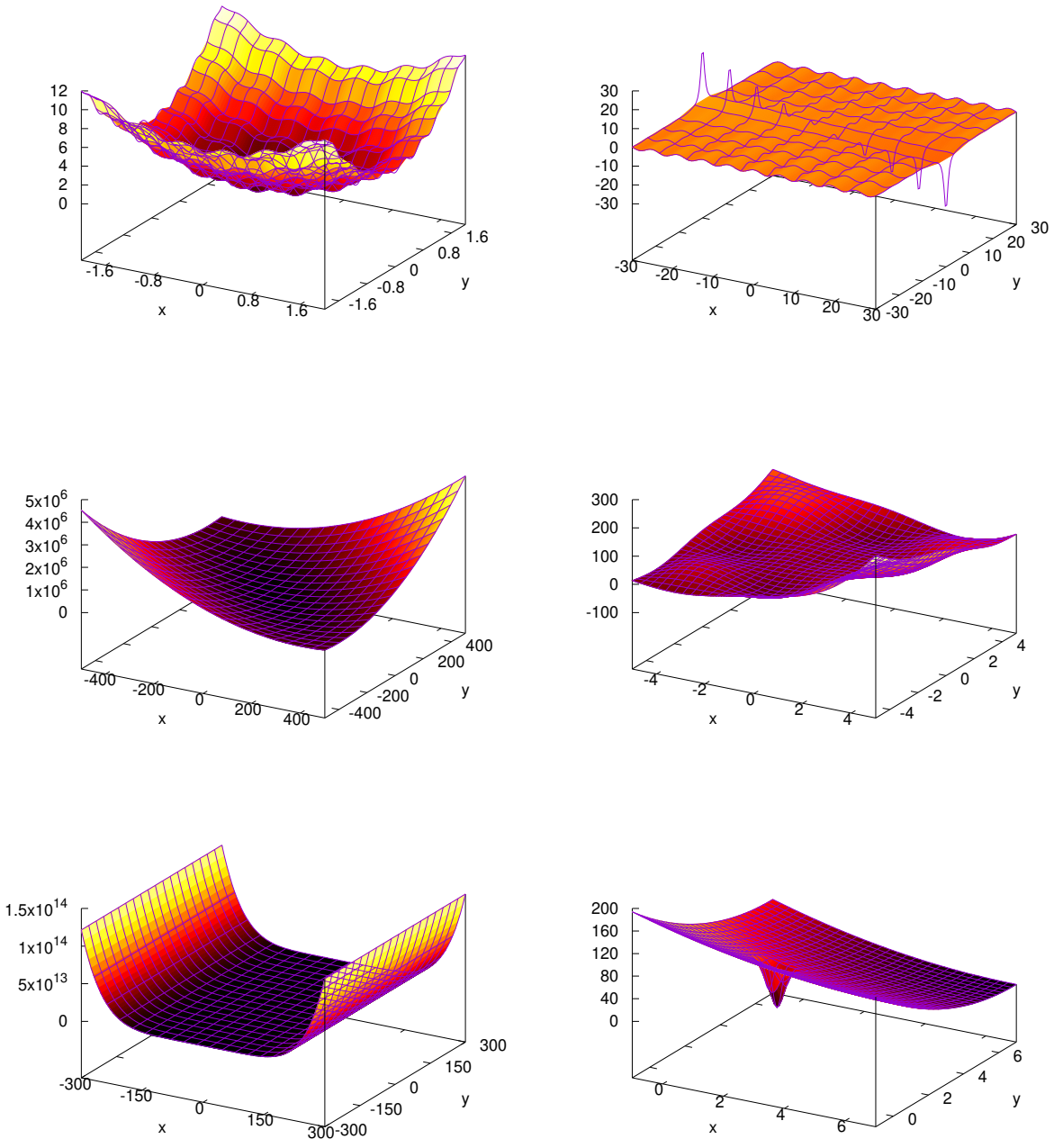

FIGURE 1.1: Various kinds of test function encountered in optimization problems **a.** convex function **b.** non convex function having significant number of equivalent minima **c.** unimodal - function having only one minimum **d.** multimodal - function having multiple minima **e.** function having large flat surface region **f.** function whose global structure provides little information about its minimum.

1. **Test Problems:** These are artificial problems designed to understand and study the performance of an algorithm in diverse circumstances often encountered in real life situations such as long narrow valley, flat surface, problems with large number of local minima etc. They are also useful to identify type of problems for which a particular method is suitable or not. Extensive lists of such problems are given in [3-5].

2. **Real Life Problems:** These optimization problems arise in tackling real life problems such as problem of finding critical points on PES mentioned earlier. Such problems often also arise in physics, chemistry, engineering, astronomy, economics etc. Usually significant amount of algebra and data processing is required before solving these problems. Real life problems are hard to find and solving them requires considerable amount of time. Often, one has to devise appropriate ways of combining more than one method to solve the problem completely.

Various properties of test problems such as convexity, modality, differentiablity and separability affect the performance of optimization methods and forms the basis for classification of these problems. Few test functions with diverse properties are shown in Fig 1.1.

1.2 A Brief Survey of Methods for Unconstrained Optimization

There exists many algorithms for solving unconstrained optimization problems and it is often the case that an algorithm is suitable for a certain class of problems. In practice, it is always good to solve a problem using more than one method or a combination of methods. Many optimization algorithms fall in one of two important categories discussed below:

1. **Search Methods:** Search methods evaluate $f(\mathbf{x})$ at a grid of points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}$... and compare them to generate the next point. These methods do not make use of gradient or curvature information and are suitable for situations where function or its derivatives are not continuous and differentiable or other methods have failed to yield a solution. Direct search using a rectangular or any other grid is a simple example.

2. **Gradient Methods:** At a point $\mathbf{x} \in \mathbb{R}^n$ the (minus of) gradient vector $\vec{\nabla} f$ points along the direction of (minimum) maximum increase of $f(\mathbf{x})$ i.e towards a (minimum) maximum. Gradient methods make use of gradient and higher order derivatives to reach towards a minimum. All gradient methods have an iterative scheme of the form[6]

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - \alpha H \nabla f|_{\mathbf{x}^{(n)}} \tag{1.3}$$

where $\mathbf{x}^{(n)}$ is the value of x after n iterations, α is a parameter and H is exact or approximate Hessian.

Complexity of optimization problems increase with dimensionality of function as it becomes more difficult to specify an optimum search direction and most optimization algorithms face the difficulty of getting trapped in a local solution. In practice, there are no efficient methods to find a global minimum. Recently, methods such as simulated anneling and genetic algorithms have been highly successful in finding global minima using stochastic approches but these methods are slow and computationally expansive. In practice, when the function landscape is smooth, gradient based methods are more efficient. Present work is motivated by the fact that use of complex variables in classical equations of motion leads to tunneling like phenomena as described in the next section and can be useful for optimization problems.

1.3 Classical Tunneling and Complex Classical Mechanics

Inspired by success of non Hermitian quantum mechanics, which studies quantum mechanics of complex non Hermitian Hamiltonians, Asiri Nanayakkara analytically studied classical dynamics of these systems and identified periodic, unbounded and chaotic trajectories in the complex plane [7]. Hermitian operators in quantum mechanics are linear operators with real eigenvalues and are associated with observable quantities such as energy of a system. Later, based on numerical studies, Bender and coworkers showed that simple complexified classical systems can exhibit interesting phenomena such as tunneling and delocalized conduction observed previously only in quantum mechanical systems [8]. Complex solutions

of classical equations of motion for various systems are studied in [7 − 13].
Since the Hamiltonian and thus energy in classical mechanics is a function of position and momentum coordinates, this implies that position and momentum are also complex numbers. Such a particle moves deterministically in complex plane and its position and momentum at any instant of time can be found by solving Hamilton's equations of motion in the complex potential. The complex potentials are obtained by analytically continuing the potential along real axis to the complex plane. In practice, whenever analytic expression of the real potential is known, this can be achieved by replacing the real variables by complex variables.

1.3.1 Some Theorems for Analytic Functions[14]

Theorem 1.1. *Let $f = u + iv$ be defined on a domain D in the complex plane where u and v are real valued. Then $f(z)$ is analytic on D if and only if $u(x, y)$ and $v(x, y)$ have continuous first order partial derivatives that satisfy the Cauchy Riemann equations i.e.*

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \qquad and \qquad \frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y} \tag{1.4}$$

Cauchy Riemann conditions are a consequence of the fact that derivative of an analytic function

$$\frac{df}{dz} = \lim_{\triangle z \to 0} \frac{f(z + \triangle z) - f(z)}{\triangle z} \tag{1.5}$$

has the same value regardless of the way in which we choose $\triangle z$. Putting $\triangle z = \triangle x$ we get,

$$\frac{df}{dz} = \frac{\partial u}{\partial x} + i\frac{\partial v}{\partial x} \tag{1.6}$$

Theorem 1.2. *If $f = u + iv$ is analytic and the functions u and v have continuous second order derivatives, then u and v are harmonic functions.*

Harmonic functions satisfies Laplace's equation, $\nabla^2\Phi = 0$. Interesting property of harmonic functions is that they do not have any local maxima or minima and value of function at any point is arithmetic average of function values at points around it.

TABLE 1.1: Real and imaginary parts of a few analytic functions of one variable.

No.	f(z)	u(x,y)	v(x,y)
1	z^2	$x^2 - y^2$	$2xy$
2	$z^4 - z^2$	$x^4 + y^4 - 6x^2y^2 - x^2 + y^2$	$4x^3y - 4xy^3 - 2xy$
3	$\cos z$	$\cos x \cosh y$	$-\sin x \sinh y$
4	$\sin z + \sin \frac{10z}{3}$	$\sin x \cosh y + \sin \frac{10x}{3} \cosh \frac{10y}{3}$	$\cos x \sinh y + \cos \frac{10x}{3} \sinh \frac{10y}{3}$

1.3.2 Classical Equations of motion in Complex plane

Consider motion of a particle in complex plane under the influence of some analytic potential $f(z)$ where $z = x+iy$ is a complex number. Classical equation of motion with one complex variable would have the form

$$m\ddot{z} = -\frac{df}{dz} \tag{1.7}$$

We use 1.6 and 1.4 to obtain

$$m\ddot{z} = -\left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right) u(x,y) \tag{1.8}$$

where $u(x,y)$ and $v(x,y)$ are real valued functions satisfying Cauchy Riemann equations. We expand $\ddot{z} = \ddot{x} + i\ddot{y}$ and compare real and imaginary parts to obtain

$$m\ddot{x} = -\frac{\partial u}{\partial x} \quad \text{and} \quad m\ddot{y} = \frac{\partial u}{\partial y} \tag{1.9}$$

Equation 1.9 implies that motion is downhill along x axis and uphill along y axis on the surface of $u(x,y)$ which means that fixed points of motion will correspond to saddle points of $u(x,y)$. On the other hand, classical motion in Euclidean plane is downhill in both the directions and fixed points corresponds to maxima and minima on potential surface. Using Cauchy Riemann equations we can also write 2.7 in terms of v alone.

$$m\ddot{x} = -\frac{\partial v}{\partial y} \quad \text{and} \quad m\ddot{y} = -\frac{\partial v}{\partial x} \tag{1.10}$$

Since f and its argument z both are complex numbers, naturally we need four dimensions to visualize any complex function which is not physically possible in our three dimensional world and there exists many techniques to visualize and study the behaviour of an analytic function. In complex classical mechanics, since

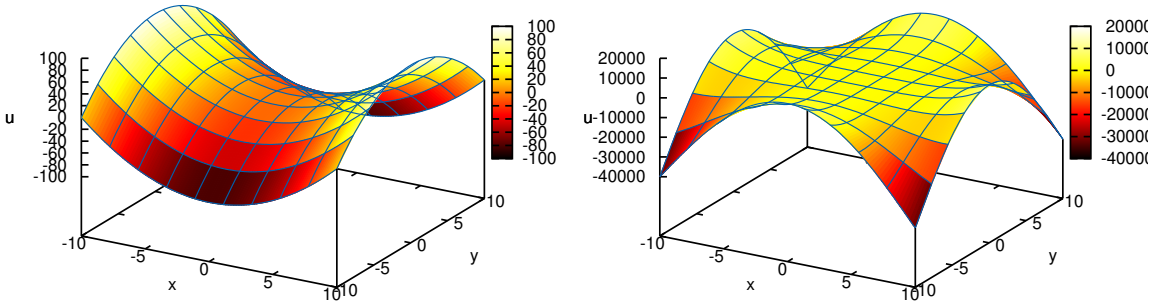

FIGURE 1.2: Real parts $(u(x, y))$ of complex potentials for one dimensional harmonic oscillator and double well potential extended in the complex plane are shown in **a** and **b** respectively.

equations of motion can be written in terms of $u(x, y)$ alone, complex classical behaviour of a system can be understood by looking at $u(x, y)$ alone.

1.3.3 Simple Complex Classical Systems

1.3.3.1 One dimensional Harmonic Oscillator

Classical motion of a particle in one dimensional oscillator

$$V(x) = \frac{1}{2}x^2 \tag{1.11}$$

is governed by the equation of motion

$$\ddot{x} = -x \tag{1.12}$$

and is well understood. For sake of simplicity, m and k are taken unity.
In the complex oscillator

$$V(z) = \frac{1}{2}z^2 \tag{1.13}$$

Equations of motion along real and imaginary axis will be

$$\ddot{x} = -x \quad \text{and} \quad \ddot{y} = -y \tag{1.14}$$

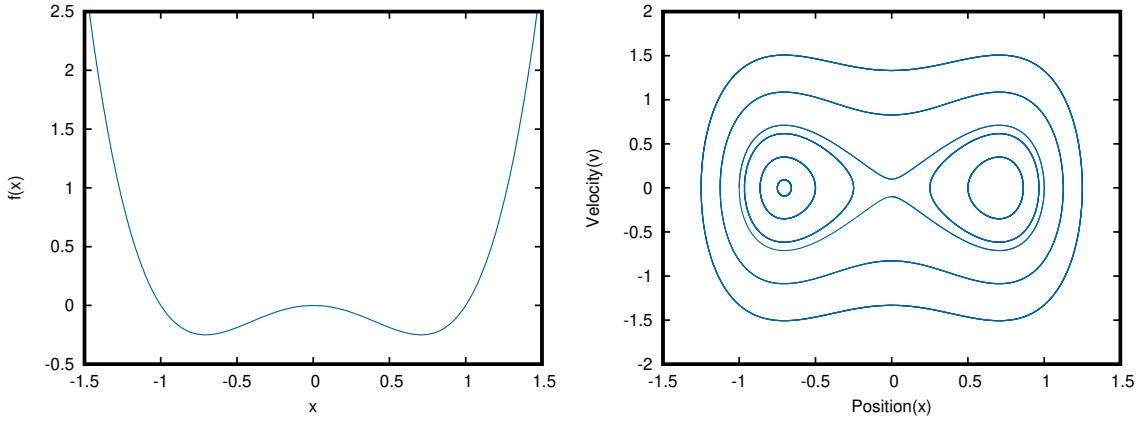

FIGURE 1.3: **a.** One dimensional double well potential $V(x) = x^4 - x^2$ and **b.** Classical trajectories in phase space for one dimensional double well potential for negative and positive energies.

which are equivalent to a two dimensional oscillator in real space.

1.3.3.2 Double Well Potential

A double well potential along one dimension

$$V(x) = x^4 - x^2 \tag{1.15}$$

is shown in Fig 1.3(a). For different energies, classical trajectories in phase space are plotted in Fig 1.3(b). A negative energy classical particle in this potential will remain bounded in one of two wells. In quantum mechanics however, there is a finite probability for a negative energy particle to be on the other side of the potential barrier.

We now consider the complex extension of this potential

$$V(z) = z^4 - z^2 \tag{1.16}$$

Equations of motion in complex plane are obtained using 1.9 and $u(x, y)$ from Table 1.1,

$$\ddot{x} = -4x^3 + 12xy^2 + 2x \quad \text{and} \quad \ddot{y} = 4y^3 - 12x^2y + 2y \tag{1.17}$$

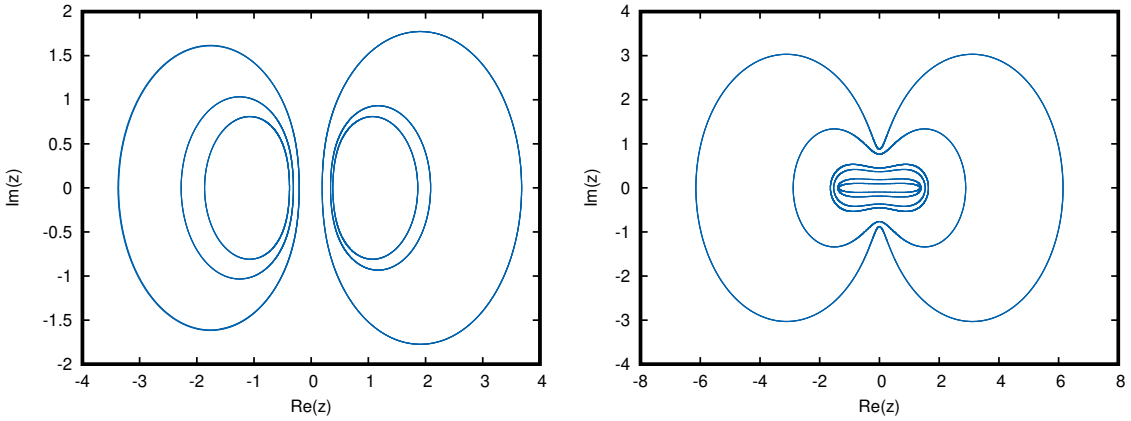

FIGURE 1.4: Closed trajectories in complex plane for real energy values in the extended double well potential. **a.** $E < 0$ **b.** $E > 0$.

Trajectories in complex plane with real energies are closed and periodic whereas those for complex energies are open and non periodic[8] and are shown in Fig 1.4 and 1.5 respectively.

Bender and Hook discussed that in the classical limit ($\hbar \to 0$), the time independent Schrodinger equation is singular because it is not possible to impose boundary or initial conditions on wavefunction at $\hbar = 0$. Because of this singular nature, some interesting features of quantum mechanics such as tunneling and discreetness of energy levels get lost abruptly in the classical limit. Using complex energy as a regulator, they showed that classical tunneling probabilities persists even when $ImE \to 0$ [9].

1.3.3.3 Periodic Potential

In numerical studies classical trajectories in complex plane of one dimensional periodic potential ($V(x) = \cos x$), it was found that for most complex energy values, complex trajectories are open and non periodic, spiralling around critical points and randomly hopping to critical points on left or right. However, there also exists a small set of energy values for which complex trajectories move unidirectionally

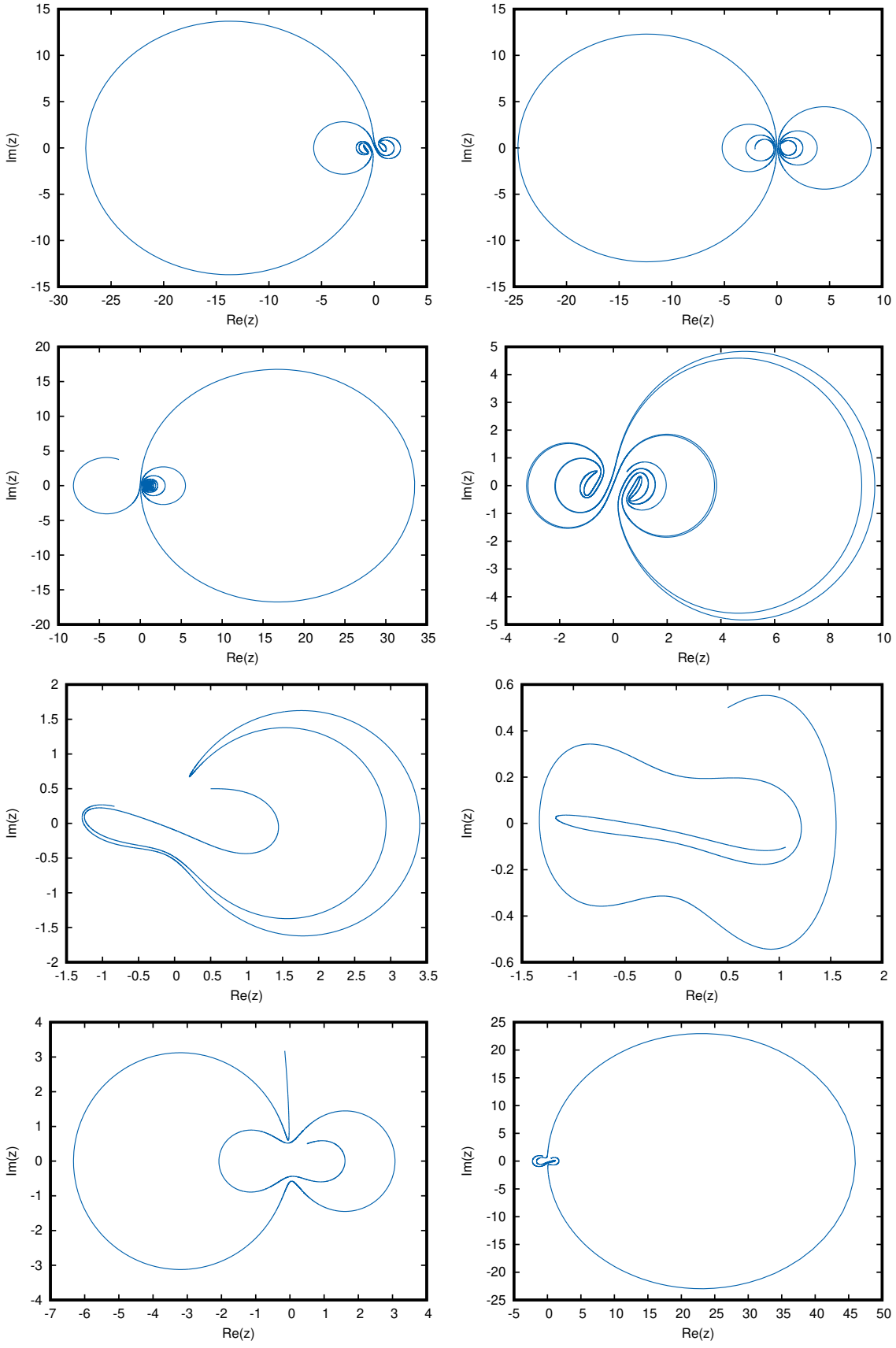

FIGURE 1.5: Trajectories in complex plane for complex energy values in extended double well potential **a.** $E = (-0.5, -0.5)$ **b.** $E = (-0.5, -0.1)$ **c.** $E = (-0.5, 0.1)$ **d.** $E = (-0.5, 0.5)$ **e.** $E = (0.5, -0.5)$ **f.** $E = (0.5, -0.1)$ **g.** $E = (0.5, 0.1)$ and **h.** $E = (0.5, 0.5)$. Trajectories for $Re(E) < 0$ and $Re(E) > 0$ differ significantly in their approach to CPs.

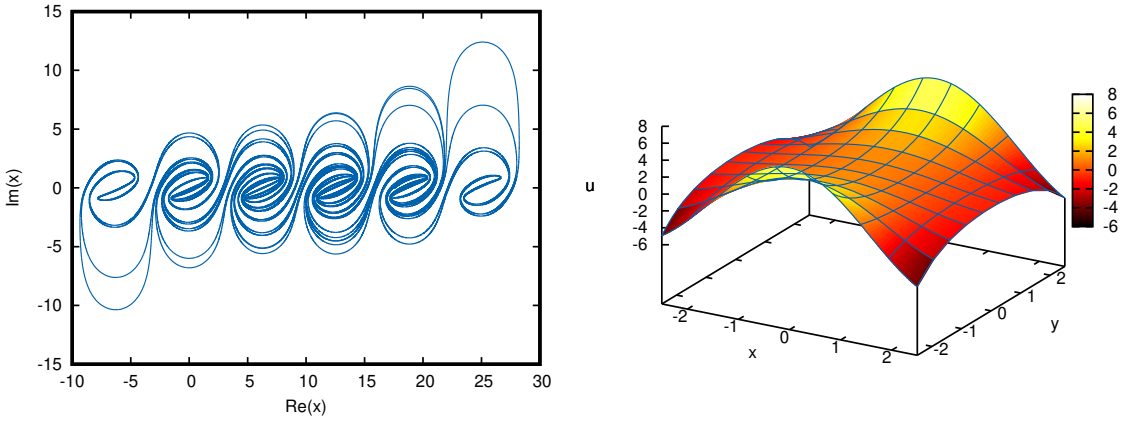

FIGURE 1.6: **a.** Complex classical trajectories with energy $(-0.5, 0.1)$ in $V(z) = \cos z$ potential and **b.** real part $(u(x, y))$ of $\cos z$ potential.

on the real axis, a behaviour similar to delocalized conduction.
In the complex potential

$$V(z) = \cos z \tag{1.18}$$

equations of motion would be

$$\ddot{x} = -\sin x \cosh y \qquad \text{and} \qquad \ddot{y} = \cos x \sinh y \tag{1.19}$$

Complex classical trajectory of a complex energy particle and $u(x, y)$ for above potential are shown in Fig 1.6.

1.4 Goals and Objectives

In last section, it was shown that Hamiltonian systems with complex variables explore classically forbidden regions in the coordinate space by taking an alternate path through complex plane. This property can be useful for optimization algorithms in escaping a local minimum and can be utilized to improve their efficiency. In the present work, a local optimizer called Newton Raphson method is used to show that other dynamical systems with complex variables also exhibit tunneling like behaviour and develop a modified Newton Raphson method for finding critical points using complex scaled variables. A set of two variable test functions is used to compare the performance of new NR method over ordinary NR method.

Chapter 2

Results and Discussion

In previous chapter, it was shown that Hamiltonian systems with complex energies spirals around fixed points of motion. Hamilton's equations constitues a second order dynamical systems and are not suitable to be used as optimizers as such because though motion of a particle approaches the CPs of potential, it does not converge to a CP. However by choosing appropriate energy values, they can be used to escape from a local minima in global optimizers. However, as discussed before other otimization methods can also benefit from extended function topology in complex space.
To test the hypotheses presented in the previous chapter, we choose a simple and efficient local optimizer called Newton Raphson method. This method provides us the best approximation to zero of gradient from a given point, by making use of curvature information and is suitable for situations where both gradient and Hessian matrices are readily available. In this chapter, we develop a modified NR method based on complex scaled variables.

2.1 Newton Raphson (NR) method for finding critical points

Newton Raphson method is a widely used first derivative method to find zeros of a function which converges to a root provided the guess point lies close to the root. This method converges quadratically to all types of critical points[15]. However

convergence is not guaranteed if the guess point lies far away from the root. This means that one needs some information about function and its CPs beforehand. Newton Raphson method is used to find critical points of a function by finding zeros of the gradient ∇f and is suitable for situations when both gradient and Hessian matrices are available. To obtain the Newton Raphson (NR) step, first ∇f is expanded as a Taylor series around the initial guess point, say $\mathbf{x}$

$$[\nabla f(\mathbf{x}+\mathbf{h})] = [\nabla f(\mathbf{x})] + [\nabla^2 f(\mathbf{x})]\mathbf{h} + \vartheta(\mathbf{h}^2) \tag{2.1}$$

and put $[\nabla f(\mathbf{x}+\mathbf{h})] = 0$ to get

$$\mathbf{h} = -[\nabla^2 f(\mathbf{x})]^{-1}[\nabla f(\mathbf{x})] \tag{2.2}$$

Here $[\nabla f(\mathbf{x})]$ and $[\nabla^2 f(\mathbf{x})]$ are gradient and Hessian matrices of f at $\mathbf{x}$. Now a better estimate to the root would be

$$\mathbf{x} = \mathbf{x}+\mathbf{h} \tag{2.3}$$

These two steps are repeated until a suitable convergence criteria is satisfied. Newton's method is one of the fastest converging algorithms with number of correct decimal places doubling each time[16].

2.2 Complex Scaled Newton Rapson (csNR) method

2.2.1 Complex Potentials

The complex potentials used in previous chapter were obtained by simply replacing the real variable by a complex variable. In this section, a parametrized way of obtaining complex potentials known as complex scaling method is described. While complex scaling, every argument of f is rotated in the complex plane by a fixed angle θ to obtain the analytically continued function[17].

$$x_i \to x_i e^{i\theta} \qquad (\theta = \text{constant}) \tag{2.4}$$

$$f(\mathbf{x}) \to f(\mathbf{x}e^{i\theta}) \tag{2.5}$$

First and second derivatives of f will transform in the complex plane

$$\frac{\partial f}{\partial x_i} = e^{i\theta}\frac{\partial f}{\partial z_i} \tag{2.6}$$

$$\frac{\partial^2 f}{\partial x_i^2} = e^{2i\theta}\frac{\partial^2 f}{\partial z_i^2} \tag{2.7}$$

The task of analytic continuation becomes more difficult when analytic expression of functions is not known as is the case with molecular potentials or PESs which are calculated on a finite grid by ab-initio quantum calculations in unscaled coordinate space. However, in quantum theory of resonances, complex scaled Hamiltonians are needed to calculate resonance widths and energies and there exist several methods for complex scaling of ab-initio molecular potentials and can be found in [17].

2.2.2 Complex Scaled Newton Raphson Method

Let $f : \mathbb{C}^n \to \mathbb{C}$ is an analytic function of complex scaled variables $\mathbf{z} = \mathbf{x}e^{i\theta}$ with continuous first and second derivatives. Since θ is constant, the complex scaled function can be expanded as a Taylor series of the form

$$f(\mathbf{x}e^{i\theta} + \mathbf{h}) = f(\mathbf{x}e^{i\theta}) + [\nabla_x f(\mathbf{x}e^{i\theta})]\mathbf{h} + [\nabla_x^2 f(\mathbf{x}e^{i\theta})]\frac{\mathbf{h}^2}{2} + \vartheta(\mathbf{h}^3) \tag{2.8}$$

where $[\nabla_x f(\mathbf{x}e^{i\theta})]$ and $[\nabla_x^2 f(\mathbf{x}e^{i\theta})]$ are gradient and Hessian matrices of partial derivatives with respect to variables x_i. Since at critical points gradient, $[\nabla_x f] = 0$, we expand $[\nabla_x f]$ as a Taylor series at some initial point $\mathbf{x}e^{i\theta} = \mathbf{z}$,

$$[\nabla_x f(\mathbf{x}e^{i\theta}+\mathbf{h})] = [\nabla_x f(\mathbf{x}e^{i\theta})] + [\nabla_x^2 f(\mathbf{x}e^{i\theta})]\mathbf{h} + \vartheta(\mathbf{h}^2) \tag{2.9}$$

using 2.6 and 2.7, this becomes,

$$[\nabla_z f(\mathbf{x}e^{i\theta}+\mathbf{h})] = [\nabla_z f(\mathbf{x}e^{i\theta})] + e^{i\theta}[\nabla_z^2 f(\mathbf{x}e^{i\theta})]\mathbf{h} + \vartheta(\mathbf{h}^2) \tag{2.10}$$

and putting $[\nabla_z f(\mathbf{x}e^{i\theta}+\mathbf{h})] = 0$, we get the Newton Raphson step,

$$\mathbf{h} = -e^{-i\theta}[\nabla_z^2 f(\mathbf{x}e^{i\theta})]^{-1}[\nabla_z f(\mathbf{x}e^{i\theta})] \tag{2.11}$$

When f is a function of more than one variables, above equation is not the best way to compute NR step since it involves inverting $n \times n$ Hessian matrix. If we can

first diagonalize the Hessian and compute its eigenvectors and eigenvalues, they can be used to simplify 2.11 without inverting the Hessian. To do that, first we write 2.11 as

$$e^{i\theta}[\nabla_z^2 f(\mathbf{x}e^{i\theta})]\mathbf{h} = -[\nabla_z f(\mathbf{x}e^{i\theta})] \tag{2.12}$$

and use the similarity transformation to diagonalize the Hessian,

$$e^{i\theta}S^{-1}[\nabla_z^2 f(\mathbf{x}e^{i\theta})]S\mathbf{h} = -S^{-1}[\nabla_z f(\mathbf{x}e^{i\theta})]S \tag{2.13}$$

Here S is the eigenvector matrix such that $S^{-1}[\nabla_z^2 f(\mathbf{x}e^{i\theta})]S = \Lambda$ is a diagonal matrix whose diagonal elements are the eigenvalues λ_is of $[\nabla_z^2 f(\mathbf{x}e^{i\theta})]$ and columns of S are eigenvectors of $[\nabla_z^2 f(\mathbf{x}e^{i\theta})]$. Since, Hessian is symmetric, $S^{-1} = S^{*T}$. Equation 2.13 now becomes

$$e^{i\theta}\Lambda\mathbf{h} = -S^{*T}[\nabla_z f(\mathbf{x}e^{i\theta})]S \tag{2.14}$$

Equation 2.14 gives us n values for h, one maximizing or minimizing along each Hessian eigendirection. Best step to move towards a critical point in n-dimensional space would be the one that minimize or maximize along all n direction which is equal to sum of each h's i.e.

$$\mathbf{h} = -e^{-i\theta}\sum_{i=1}^{n}\frac{(u_i^{*T}.[\nabla_z f(\mathbf{x}e^{i\theta})])u_i}{\lambda_i} \tag{2.15}$$

For real functions, an equivalent expression of NR step is given in [2,14].

2.3 Dynamics of csNR in Complex Plane

In previous section while deriving NR step for complex scaled functions we assumed that we have a complex analytic function $f : \mathbb{C}^n \to \mathbb{C}$ of complex scaled variables which is not the case in practice. In other words, we assumed that we have θ such that $f(\mathbf{x}e^{i\theta})$ is an analytic function. To obtain the scaling parameter θ, we studied the dynamics of csNR in complex plane for functions of one variable listed in Table 2.1 and then decide to obtain θ using a simple search strategy.

Beginning from same initial point on real line, trajectories of csNR in complex plane for different values of complex scaling parameter θ are shown in Fig 2.1 for functions of one variable listed in Table 2. We found that as θ is increased,

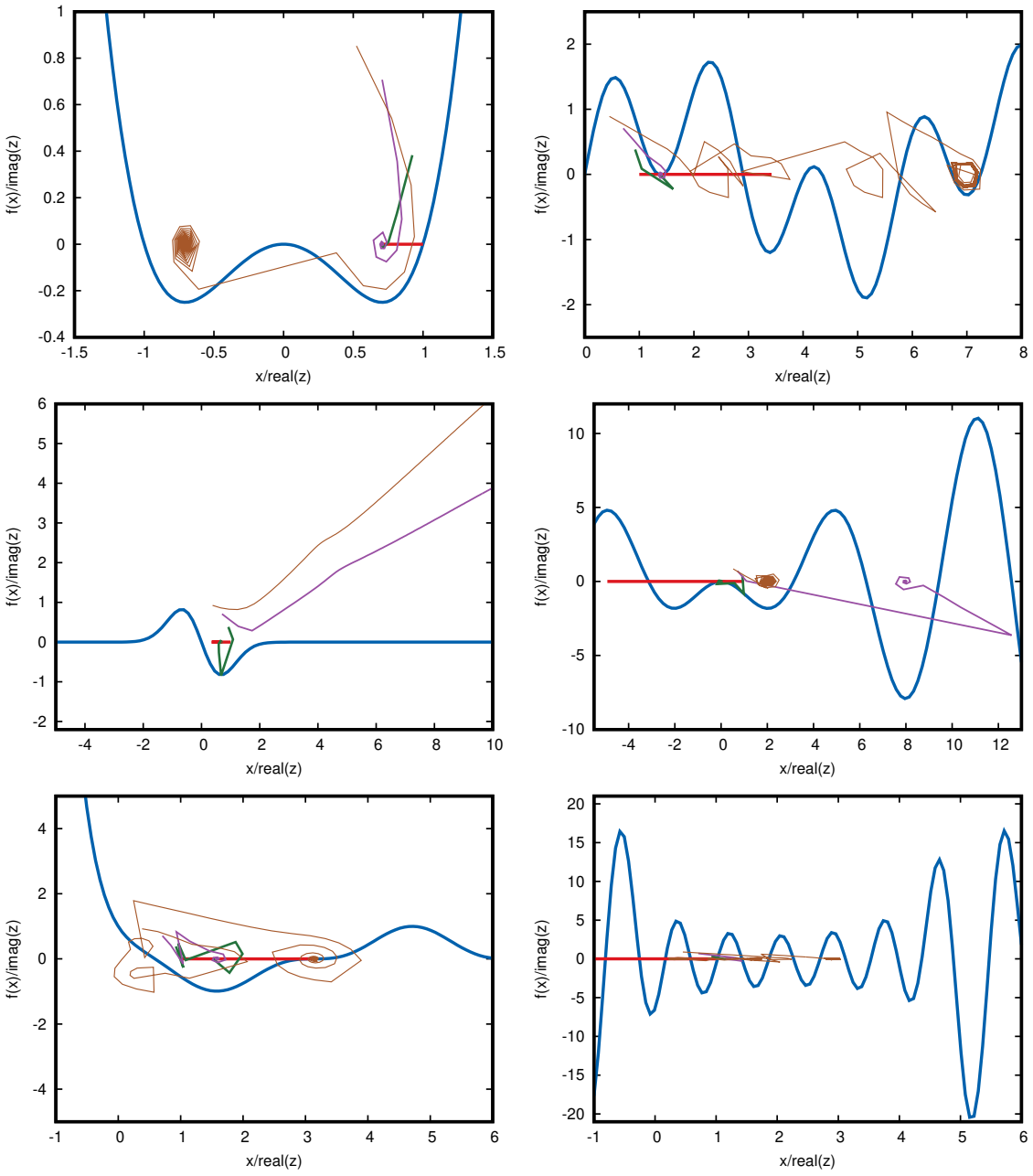

FIGURE 2.1: Trajectories of csNR in complex plane for one variable test functions listed Table 2.1 for different values of θ.. Red, $\theta = 0$. Green, $\theta = \frac{\pi}{8}$. Violet, $\theta = \frac{\pi}{4}$. Brown, $\theta = \frac{3\pi}{8}$. As θ is increased, csNR approaches more critical points in the nearby region and converges to different CPs for different θ.

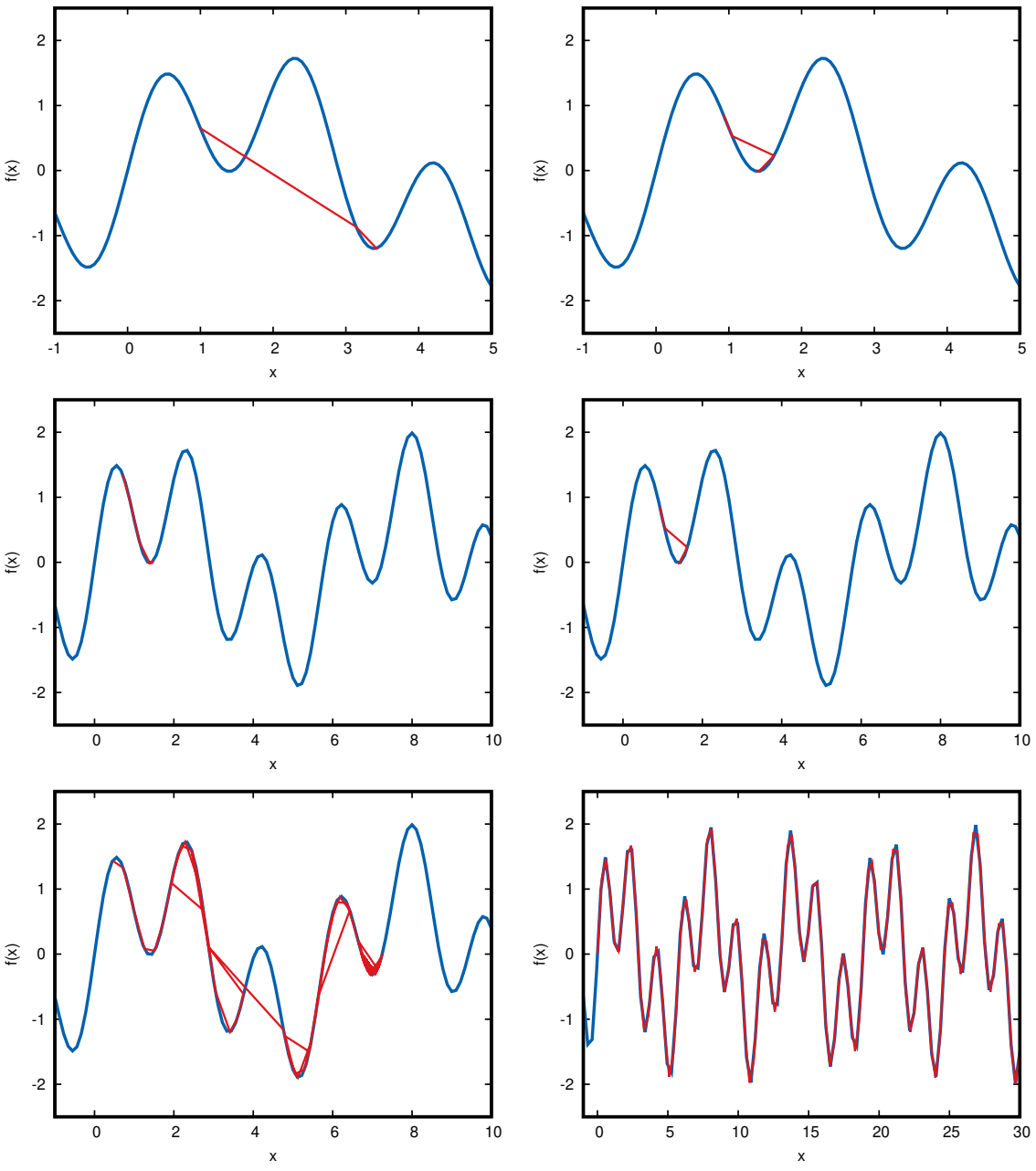

FIGURE 2.2: Trajectories of csNR in real function space for $f(x) = \sin x + \sin \frac{10x}{3}$ for different values of θ showing dependence of convergence of method on the scaling parameter θ. **a.** $\theta = 0$. **b.** $\theta = \frac{\pi}{8}$. **c.** $\theta = \frac{\pi}{4}$. **d.** $\theta = \frac{13\pi}{40}$. **e.** $\theta = \frac{14\pi}{40}$. **f.** $\theta = \frac{\pi}{2}$. As θ is increased, csNR takes more steps to converge to a CP.

TABLE 2.1: One variable test functions used to study the behaviour of csNR in complex plane.

No.	f(x)
1	$x^4 - x^2$
2	$\sin x + \sin \frac{10x}{3}$
3	$-(x + \sin x)e^{-x^2}$
4	$-x \sin x$
5	$e^{-3x} - \sin^3 x$
6	$\sum_{k=1}^{6} k \sin((k+1)x + k)$

csNR move in the complex plane around a critical point eventually converging or escaping towards another nearby critical points.

A good comparison of csNR with ordinary NR method is obtained from Fig 2.2 where real parts of complex variable z is plotted against the real function value at that point for $f(x) = \sin x + \sin \frac{10x}{3}$ for different values of θ. These trajectories show that as $\theta \to \frac{\pi}{2}$, csNR takes more steps to converge and the extent of tunneling increases and at $\theta = \frac{\pi}{2}$ it goes all the way to infinity which is in accordance with Eq 2.15. Similar behaviour was observed for other functions also.

Figure 2 and 3 show that as θ is increased, extent of tunneling increases taking the method to previously unexplored regions in coordinate space, but number of steps to converge to a root also increases. This implies that there is a trade off between tunneling and convergence in csNR and we can not choose θ arbitrarily. In the optimization algorithm we developed, for different θ values, starting from a small number of initial random points, we retained θ returning largest number of critical point as optimum value of θ for that function.

2.4 Comparing csNR with ordinary NR method

In previous section, it was shown that Newton Raphson method can be used to find critical points of complex scaled functions. However, in deriving equation 2.15, it was assumed that we have an analytic function $f : \mathbb{C}^n \to \mathbb{C}$ of complex scaled variables $\mathbf{z} = \mathbf{x}e^{i\theta}$ or in other words we have a good value of scaling parameter θ. We first study the dynamical behaviour of complex scaled Newton Raphson method in complex plane and come up with a way of obtaining the scaling parameter for csNR and than test and compare the csNR method with ordinary NR method for

TABLE 2.2: Two variables test function for optimization along with θ_{opt} for each function. θ_{opt} for each function was determined by running csNR subroutine in Appendix A for θ values in $(0, \frac{\pi}{50}, \frac{\pi}{25}, .., \frac{\pi}{2})$ for 1000 initial points chosen randomly in $x_i \in [-100, 100]$.

No.	**f(x,y)**	θ_{opt}
1	$\cos x \sin y - \frac{x}{y^2+1}$	0.000
2	$\sin x e^{(1-\cos y)^2} + \cos y e^{(1-\sin x)^2} + (x-y)^2$	0.000
3	$x^2 + 2y^2 - 0.3\cos(3\pi x) - 0.4\cos(4\pi y) + 0.7$	0.063
4	$x^2 + 2y^2 - 0.12\cos(3\pi x)\cos(4\pi y) + 0.3$	0.126
5	$x^2 + 2y^2 - 0.3\cos(3\pi x + 4\pi y) + 0.3$	0.000
6	$(y - \frac{5.1x^2}{4\pi^2} + \frac{5x}{\pi} - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x + 10$	1.005
7	$(y - \frac{5.1x^2}{4\pi^2} + \frac{5x}{\pi} - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x \cos y \ln(x^2+y^2+1) + 10$	0.126
8	$2x^2 - 1.05x^4 + \frac{x^6}{6} + xy + y^2$	1.005
9	$(4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + 4(y^2-1)y^2$	1.005
10	$-\frac{0.001}{0.001^2+(x-0.4y-0.1)^2} - \frac{0.001}{0.001^2+(2x+y-1.5)^2}$	1.382
11	$-\frac{0.001}{0.001^2+(x^2+y^2-1)^2} - \frac{0.001}{0.001^2+(x^2+y^2-0.5)^2} - \frac{0.001}{0.001^2+(x^2-y^2)^2}$	1.382
12	$x^2 - 12x + 11 + 10\cos(\frac{\pi x}{2}) + 8\sin(\frac{5\pi x}{2}) - \sqrt{\frac{1}{5}}e^{-0.5(y-0.5)^2}$	0.817
13	$[1 - \lvert\frac{\sin(\pi(x-2)\sin(\pi(y-2))}{\pi^2(x-2)(y-2)}\rvert^5][2 + (x-7)^2 + 2(y-7)^2]$	0.440
14	$10^5 x^2 + y^2 - (x^2+y^2) + 10^{-5}(x^2+y^2)^4$	1.005
15	$-\cos x \cos y e^{-(x-\pi)^2-(y-\pi)^2}$	0.754
16	$x^2 + y^2 + 25(\sin^2 x + \sin^2 y)$	0.063
17	$(x - 13 + ((5-y)y - 2)y)^2 + (x - 29 + ((y+1)y - 14)y)^2$	0.880
18	$[1 + (x+y+1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2)]$ $\times[30 + (2x-3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2)]$	1.005
19	$(x^2 + y - 11)^2 + (x + y^2 - 7)^2$	1.005
20	$[\sin(x-y)\sin(x+y)]^2/\sqrt{x^2+y^2}$	0.000
21	$\sin(x+y) + (x-y)^2 - 1.5x + 2.5y + 1$	0.000
22	$\sqrt{\lvert\cos\sqrt{\lvert x^2+y^2\rvert}\rvert} + 0.01(x+y)$	1.382
23	$\sqrt{\lvert\sin\sqrt{\lvert x^2+y^2\rvert}\rvert} + 0.01(x+y)$	1.382
24	$\{\sin^2[(\cos x + \cos y)^2] + \cos^2[(\sin x + \sin y)] + x\}^2 + 0.01(x+y)$	0.000
25	$-ln\{\{\sin^2[(\cos x + \cos y)^2] + \cos^2[(\sin x + \sin y)] + x\}^2\} +$ $0.01[(x-1)^2 + (y-1)^2]$	0.000
26	$0.001\Big\{\lvert x^{10} - 20x^9 + 180x^8 - 960x^7 + 3360x^6 - 8064x^5 + 1334x^4$ $-15360x^3 + 11520x^2 - 5120x + 2624\rvert \times \lvert y^4 + 12y^3 + 54y^2 + 108y + 81\rvert\Big\}^2$	0.000
27	$\cos x^2 + \sin y^2$	0.000
28	$-\exp\{\Big\lvert\cos x \cos y \exp\{\lvert 1 - \frac{\sqrt{x^2+y^2}}{\pi}\rvert\}\Big\rvert^{-1}\}$	0.063
29	$1 + \sin^2 x + \sin^2 y - 0.1e^{-x^2-y^2}$	0.000

30	$100(y-x)^2+6\left\lceil 6.4(y-0.5)^2-x-0.6\right\rceil^2$	1.005
31	$(2x^3y-y^3)^2+(6x-y^2+y)^2$	0.942
32	$74+100(y-x^2)^2+(1-x)^2-400\exp\{-\frac{(x+1)^2+(y+1)^2}{0.1}\}$	0.000
33	$-xy(72-2x-2y)$	1.382
34	$-\left\lvert\cos x\cos y\exp\{\lvert 1-\frac{\sqrt{x^2+y^2}}{\pi}\rvert\}\right\rvert^{-1}$	0.000

TABLE 2.3: Optimization results for two dimensional test functions listed in Table 2.2. Number of distinct critical points found for $\theta=\theta_{opt}$ and $\theta=0$ are respectively shown in columns 2 and 5. CPU time and number of total number of steps in two cases are listed in columns 3,4 and 6,7. In columns 8,9,10 are given percentage changes in these quantities. For functions not listed in this table but listed in Table 2.2, θ_{opt} is zero.

Sr.	$\theta=\theta_{opt}$			$\theta=0.0$			% change in		
No.	NCP	TIME	STEPS	NCP	TIME	STEPS	NCP	TIME	STEPS
3	3227	1.324	164359	2828	1.499	210654	14.11	-11.64	-21.98
4	4965	6.947	789936	4502	4.839	564774	10.28	43.57	39.87
6	174	3.854	566865	115	0.195	27844	51.3	1878.26	1935.86
7	3065	1.612	143734	2629	2.454	240598	16.58	-34.32	-40.26
8	10	9.355	1723771	5	0.484	96450	100	1833.11	1687.22
9	31	6.502	1219397	8	0.502	100647	287.5	1195.18	1111.56
10	171	6.96	1252091	94	4.306	841535	81.91	61.62	48.79
11	123	12.714	2146247	124	6.165	1121291	-0.81	106.21	91.41
12	934	1.408	215282	74	0.619	108797	1162.16	127.59	97.87
13	278	27.43	2823928	1	0.089	10004	27700	30745.23	28127.99
14	18	9.319	1702973	5	0.375	75661	260	2381.85	2150.79
15	569	27.635	2636703	141	0.227	29705	303.55	12077.16	8776.29
16	3221	1.312	155478	2733	9.806	1303796	17.86	-86.62	-88.07
17	8	2.732	530238	3	0.439	89437	166.67	522.66	492.86
18	36	6.931	1222514	13	0.909	167076	176.92	662.44	631.71
19	28	6.615	1307434	8	0.304	61356	250	2072.77	2030.9
22	1641	22.203	3738494	555	26.753	4703671	195.68	-17.01	-20.52
23	1578	22.548	3780624	528	27.027	4720017	198.86	-16.57	-19.9
28	108	0.782	76350	55	0.681	74674	96.36	14.84	2.24
30	19	5.481	1072393	5	0.431	88619	280	1172.99	1110.12
31	45	17.129	3006828	17	1.447	273801	164.71	1083.45	998.18
33	107	10.763	2118632	4	0.176	35422	2575	6029.34	5881.12

TABLE 2.4: Comparison of performance of csNR over conventional NR method for quantities shown in Table 2.3

	NCP	STEPS	CPU TIME
Increased	21	17	16
Decreased	0	5	5
Remains same	13	12	13

finding critical points.

To test and compare the performance of csNR method over conventional NR method we selected a set of two variable continuous, differentiable and multimodel test functions suitable for Newton Raphson method from Jamil and Yang's extensive list of test functions for unconstrained optimization [3]. These test functions are listed in Table 2.2 alongwith θ_{opt} for each function. We found that θ_{opt} equals zero for 12 functions which means that for these functions complex scaling does not offer any advantage and instead csNR would end up using extra storage for complex numbers.

For remaining 22 functions test results obtained in the two cases are shown in Table 2.3 and conclusions are briefly summarized in Table 2.4. We see that except in one case, csNR method found more number of critical points however often taking more steps to reach a critical point and requiring more computational resources than ordinary NR method. Though this is not always true and there are instances where an increase in critical points is observed alongwith a decrease in number of steps as we can see from Tables 2.3.

2.5 Conclusions and Future Possibilities

From previous studies of complex classical systems and our study of csNR method, we conclude that use of complex variables provides extra degrees of freedom to the dynamical systems which allow it to explore previously unexplored regions in real coordinate space by making use of extended complex space. We showed that for csNR method the extent of this exploration depends directly on the complex scaling parameter θ. This happens at the cost of convergence which depends inversely on θ and becomes diverging when $\theta = \frac{\pi}{2}$. Test results show that csNR finds more CPs for almost two-third of continuous, differentiable and multimodel

test problems of two variables.

Future work plan is given below:

- Devise a global optimizer using search strategies and memory structures.
- Develop complex potential energy surface for ethane molecule and optimize it using csNR.

Appendices

Appendix A

Fortran90 Program for comparing csNR with NR

This program is devided into four parts briefly described below

1. **program nr2varcs_main**: It first evaluates θ_{opt} for a function and then finds critical points using the two methods.
2. **subroutine csNR**: This is a the complex scaled Newton Raphson subroutine which takes θ and initial guess and converges to a critical point. It calls subroutine clsfiy to determine the type of critical point.
3. **subroutine sort**: It stores distinct critical points based on their type.
4. **subroutine clsfiy**: This subroutine determines the type of critical point by evaluating hessian matrix and diagonalizing it.

```
program nr2varcs_main

implicit none
integer,parameter::n0=2
real*8,parameter::pi=4.d0*atan(1.d0)
complex*16,parameter::io=complex(0.d0,1.d0)
integer::ii,i,j,keyn, numb, nmax, nmin, nsdd, numcp
integer:: step, topt, tzero, numcp0, n, numcpo
real*8:: theta,optheta,x(n0,1),y(n0,1)
real*8::t1,t2,t3,perct,percp,percs,xr(n0,1)
real*8,allocatable:: mxima(:,:), mnima(:,:), sddle(:,:)
```

```
character(len=200)::fmt1
fmt1 = '(1f7.3,a1,i5,a1,1f7.3,a1,i9,a1,i5,a1,1f7.3,a1,i9,a1,1f8.2,a1,1f8.2,a1,1f8.2,a2)'

open(121,file='mnima')
open(122,file='mxima')
open(123,file='sddle')
open(124,file='best')
OPEN(2,file='thetas')
open(3,file='output')


!%%%%%FINDING THETA_OPTIMUM%%%%%
numcpo = 0
do 20 i = 0,24
        theta = i*pi/50
        nmin = 0     !no of minima found
        nmax = 0     !no of maxima found
        nsdd = 0    !no of sddles found
        numb = 0     !no of initial points that converged
        n = 1000
        allocate(mxima(n,n0), mnima(n,n0), sddle(n,n0))
        do 21 j = 1,n
           do 22 ii = 1,n0
                 x(ii,1) = -100+200*rand()  !random initial guess
            22 enddo
            call csNR(x,theta,y,keyn,step)  !calling csNR subroutine.
            if (keyn .ne. 2) then    !csNR converged.
                  numb = numb + 1
                  call sort(y,keyn,mnima,mxima,sddle,nmin,nmax,nsdd)
            endif
        21 enddo
        numcp = nmin + nmax + nsdd    !no of distinct CPs found
        deallocate(mxima, mnima, sddle)
        write(2,'(1F11.4,A1,i8,a1,i8,a1,i8,a1,i8,a1,i8,a2)') theta, '&', nmin, '&', &
         & nmax, '&', nsdd , '&', numcp, '&', numb, '\\'
        if (numcp .gt. numcpo) then
                numcpo = numcp
                optheta = theta
        endif
20 enddo
write(124,'(1f8.3)') optheta


!%%%%%COMPARING csNR AND NR%%%%%
```

```
call cpu_time(t1)
topt = 0     !total steps in csNR
tzero = 0     !total steps in NR
do i = 0,0
        theta = optheta
        nmin = 0
        nmax = 0
        nsdd = 0
        numb = 0
        n = 5000
        allocate(mxima(n,n0), mnima(n,n0), sddle(n,n0))
        do j = 1,n
                do ii = 1,n0
                        x(ii,1) = -100+200*rand()
                enddo
                call nrcs2var(x,theta,y,keyn,step)
                topt = topt + step
                if (keyn .ne. 2) then
                        numb = numb + 1
                        call sort(y,keyn,mnima,mxima,sddle,nmin,nmax,nsdd)
                endif
        enddo
        numcp = nmin + nmax + nsdd
        deallocate(mxima, mnima, sddle)
enddo

call cpu_time(t2)

do i = 0,0
        theta = 0.d0
        nmin = 0
        nmax = 0
        nsdd = 0
        numb = 0
        n = 5000
        allocate(mxima(n,n0), mnima(n,n0), sddle(n,n0))
        do j = 1,n
         do ii = 1,n0
                 x(ii,1) = -100+200*rand()
                enddo
                call nrcs2var(x,theta,y,keyn,step)
                tzero = tzero + step
                if (keyn .ne. 2) then
                        numb = numb + 1
```

```
                        call sort(y,keyn,mnima,mxima,sddle,nmin,nmax,nsdd)
                endif
        enddo
        numcp0 = nmin + nmax + nsdd
        deallocate(mxima, mnima, sddle)
enddo
call cpu_time(t3)

percp = 100.d0*(numcp-numcp0)/numcp0
perct = 100.d0*((t2-t1)-(t3-t2))/(t3-t2)
percs = 100.d0*(topt-tzero)/tzero

write(3,fmt1) optheta,'&',numcp,'&',t2-t1,'&',topt,'&',numcp0,'&',t3-t2,'&', &
     & tzero,'&',percp,'&',perct,'&',percs,'\\'

end program nr2varcs
!========================================================================

SUBROUTINE csNR(xr,theta,y,keyn,step)
!Newton Raphson routine for two dimensional complex scaled functions.

implicit none
integer,parameter::n0=2,n=1000
integer :: ii, i1, i2, info, keyn, i, k1, key1, step
real*8,parameter::pi=4.d0*atan(1.0d0),eps=1.0d-4,h=0.001
real*8::xr(n0,1),theta,rwork(2*n0),rex(n0,1),fr1,y(n0,1),eval(n0),hugee,abdx(n0,1),magdx
complex*16,parameter::io=complex(0.d0,1.d0),hc=complex(0.001,0.001)
complex*16::fc1,x(n0,1),h1(n0,1),h2(n0,1),h3(n0,1),grd(n0,1),hess(n0,n0),w(n0),dum(1,1)
complex*16::vl(n0,n0),vr(n0,n0),work(4*n0),dx(n0,1),dummy(n0,1),rvec(n0,1),lvec(1,n0)
character(len=30)::format1
format1 = "(1f10.8,2f16.6,i5,11f16.6,)"

open(19,file='path')
open(20,file='file')

hugee = huge(0.0d0)
keyn = 0
do 10 ii=1,n0
   h1(ii,1) = 0.d0
   h2(ii,1) = 0.d0
   h3(ii,1) = 0.d0
10 enddo

do ii=1,n0
```

```
   x(ii,1) = cdexp(io*theta)*xr(ii,1)
enddo

do 11 i=1,n
   step = i
!EVALUATING GRADIENT AND HESSIAN MATRICES.
   do 12 i1 = 1,n0
      h1(i1,1) = hc
      grd(i1,1) = cdexp(io*theta)*(fc1(x+h1)-fc1(x-h1))/(2.d0*hc)
      !print*, 'grad(i)', i1, grd(i1,1)
      h1(i1,1) = 0.d0
      do 13 i2 = 1,n0
         h2(i1,1) = hc
         h3(i2,1) = hc
         hess(i1,i2)=((fc1(x+h2+h3)-fc1(x-h2+h3)-fc1(x+h2-h3)+fc1(x-h2-h3))/(4.d0*hc*hc))*
             & cdexp(2.d0*io*theta)
         !print*, 'hess', i1,i2, real(hess(i1,i2)), aimag(hess(i1,i2))
         h2(i1,1) = 0.d0
         h3(i2,1) = 0.d0
      13 enddo
   12 enddo

!CHECKING HESSIAN FOR NAN AND INFINITY
  do i1 = 1,n0
    do i2 = 1,n0
      if(isnan(real(hess(i1,i2))).or.abs(real(hess(i1,i2))).gt.hugee)then
          k1 = 0
          exit
      else if(isnan(aimag(hess(i1,i2))).or.abs(aimag(hess(i1,i2))).gt.hugee)then
          k1 = 0
          exit
      else
          k1 = 1
      endif
      enddo
      if (k1 .eq. 0) exit
   enddo
   if (k1 .eq. 0) then
      keyn = 2   !METHOD DIVERGED
      exit
   endif

!DIAGONALIZING HESSIAN
   call zgeev('v', 'v', n0, hess, n0, w, vl, n0, vr, n0, work, 4*n0, rwork, info)
```

```
   if (info .ne. 0 ) then
      print*, 'ZGEEV failed to find all eigenvalues and eigenvectors'
      exit
   endif

!EVALUATING NR STEP
   do 14 ii=1,n0
      dx(ii,1) = 0.d0
   14 enddo
   do 15 ii=1,n0
      do i1 = 1,n0
         dummy(i1,1) = conjg(vl(i1,ii))
         rvec(i1,1) = vr(i1,ii)
      enddo
      lvec = transpose(dummy)
      dum = (1/w(ii))*matmul(lvec,grd)
      dx = dx + dum(1,1)*rvec
   15 enddo

!EVALUATING ||dz||
   magdx = 0.d0
   do ii=1,n0
        abdx(ii,1) = real(dx(ii,1))**2 + aimag(dx(ii,1))**2
        magdx = magdx + abdx(ii,1)
   enddo
   magdx = sqrt(magdx)
   do 17 ii=1,n0
        if (magdx .lt. eps) then
             key1 = 1   !METHOD CONVERGED
        else
             key1 = 0   !NOT CONVERGED YET
             exit
        endif
   17 enddo

!DETERMINING NATURE OF CP IF CONVERGED
   if (key1 .eq. 1) then
        do ii =1,n0
             y(ii,1) = real(x(ii,1))
        enddo
        call clsfiy(y,eval,keyn)    !CALLING CLASSIFIER.
        exit
   else
    x = x - dx
```

```
   endif
11 enddo

end subroutine nrcs2var
!=====================================================================
subroutine sort(y,keyn,mnima,mxima,sddle,nmin,nmax,nsdd)
        !b_k sept 14,2017
        implicit none
        integer,parameter::n0=2
        integer::nmin,nmax,nsdd,k1,ii,i,keyn
        real*8,parameter::eps=1.0d-4
        real*8::y(n0,1),mxima(1000,2),mnima(1000,2),sddle(1000,2),ss,fr1

        if (keyn .eq. 1) then
                if (nmin .eq. 0) then
                        k1 = 1
                else
                        do i = 1,nmin
                               do ii = 1,n0
                                       ss = mnima(i,ii) - y(ii,1)
                                       if (abs(ss) .lt. eps) then
                                              k1 = 0
                                       else
                                              k1 = 1
                                              exit
                                       endif
                               enddo
                               if ( k1 .eq. 0) exit
                        enddo
                endif
                if ( k1 .eq. 1) then
                        nmin = nmin + 1
                        do ii = 1,n0
                                mnima(nmin,ii) = y(ii,1)
                        enddo
                        write(121,'(3f15.5)') (y(ii,1),ii=1,n0),fr1(y)
                endif
        elseif (keyn .eq. -1) then
                if (nmax .eq. 0) then
                        k1 = 1
                else
                        do i = 1,nmax
                               do ii = 1,n0
                                       ss = mxima(i,ii) - y(ii,1)
```

```
                                if (abs(ss) .lt. eps) then
                                        k1 = 0
                                else
                                        k1 = 1
                                        exit
                                endif
                        enddo
                        if ( k1 .eq. 0) exit
                enddo
        endif
        if ( k1 .eq. 1) then
                nmax = nmax + 1
                do ii = 1,n0
                        mxima(nmax,ii) = y(ii,1)
                enddo
                write(122,'(3f15.5)') (y(ii,1),ii=1,n0),fr1(y)
        endif
        endif
    elseif (keyn .eq. 0) then
        if (nsdd .eq. 0) then
                k1 = 1
        else
                do i = 1,nsdd
                        do ii = 1,n0
                                ss = sddle(i,ii) - y(ii,1)
                                if (abs(ss) .lt. eps) then
                                        k1 = 0
                                else
                                        k1 = 1
                                        exit
                                endif
                        enddo
                        if ( k1 .eq. 0) exit
                enddo
        endif
        if ( k1 .eq. 1) then
                nsdd = nsdd + 1
                do ii = 1,n0
                        sddle(nsdd,ii) = y(ii,1)
                enddo
                write(123,'(3f15.5)') (y(ii,1),ii=1,n0),fr1(y)
        endif
    endif
end subroutine sort
```

```
!================================================
subroutine clsfiy(x,eval,keyn)
!EVALUATES AND DIAGNOLIZE THE HESSIAN MATRIX AT THE CRITICAL POINT.
        implicit none
        integer,parameter::n0=2
        integer::ii,i1,i2,info,keyn
        real*8,parameter::h=0.001
        real*8::fr1,x(n0,1),eval(n0),h2(n0,1),h3(n0,1),hess(n0,n0),work(3*n0-1)

        do 112 i1 = 1,n0
            do 113 i2 = i1,n0
                 h2(i1,1) = h
                 h3(i2,1) = h
                 hess(i1,i2) = (fr1(x+h2+h3)-fr1(x-h2+h3)-fr1(x+h2-h3)+fr1(x-h2-h3)) &
                       &/(4.d0*h*h)
                 h2(i1,1) = 0.d0
                 h3(i2,1) = 0.d0
            113 enddo
        112 enddo

        CALL DSYEV('V','U', N0, HESS, N0, eval, WORK ,3*N0-1, INFO)

        if (eval(1) .gt. 0) then
                 if (eval(2) .gt. 0) then
                          keyn = 1                !minima
                 else
                          keyn = 0                !saddle
                 endif
        else
                 if (eval(2) .gt. 0) then
                          keyn = 0                !saddle
                 else
                          keyn = -1               !maxima
                 endif
        endif

end subroutine clsfiy
!================================================
```

Bibliography

[1] A. Banerjee, N. Adams and J. Simons. *Search for stationary points on surfaces.* J. Phys. Chem. 89, 52-57 (1985)

[2] P. Balanarayan and S.R. Gadre. *Topography of molecular scalar fields. I. Algorithm and Poincare-Hopf relation.* J. Chem. Phys. 119, 5037(2003)

[3] M. Jamil and Xin-She Yang. *A literature survey of benchmark functions for global optimization.* Int. Journal of Mathematical Modelling and Numerical Optimisation, 4, 150-194(2013)

[4] N. Andrei. *An Unconstrained Optimization Test Functions Collection* Adv. Modelling and Optimization, 10, 147-161(2008).

[5] Andrea Gavana. *1-D Test Functions.* [www.infinity77.net] (2017)

[6] E. V. Krishnamurthy and S K Sen. *Numerical Algorithms* Book published by AEWP, New Delhi 1986

[7] Asiri Nanayakkara. *Classical trajectories of 1D complex non-Hermitian Hamiltonian systems.* J. Phys. A: Math Gen. 37(2004) 4321-4334.

[8] C. M. Bender, D. C. Brody and D. W. Hook. *Quantum effects in classical systems having complex energy.* Journal of Physics A: Mathematical and Theoretical 41, 352003 (2008)

[9] C. M. Bender and D. W. Hook. *Quantum tunneling as a classical anomaly.* Journal of Physics A: Mathematical and Theoretical 44, 372001 (2011)

[10] C. M. Bender, D. D. Holm and D. W. Hook. *Complex Trajectories of a Simple Pendulum.* Journal of Physics A: Mathematical and Theoretical 40, F81-F89 (2007)

[11] C.M.Bender, D.W.Hook and K. Kooner. *Complex Elliptic Pendulum.* [arXiv:1001.0131v1]

[12] B. P. Mandal and S. S. Mahajan. *Complex classical mechanics of a QES potential.* [arXiv:1312.0757]

[13] A.G.Anderson, C.M.Bender and U.I.Morone. *Periodic orbits for classical particles having complex energy.* [arXiv:1102.4822v1]

[14] T. W. Gamelin. *Complex Analysis.* Book by Springer-Verlag, New York 2001

[15] P.L.A. Popelier. *A robust algorithm to locate automatically all types of critical points in the charge density and its Laplacian.* Chem. Phys. Lett. 228, 160-164 (1994)

[16] C. J. Cerjan and W. H. Miller. *On finding transition states.* J. Chem. Phys. 75, 2800 (1981)

[17] Nimrod Moiseyev. *Quantum theory of resonances: calculating energies, widths and cross-sections by complex scaling.* Physics Reports 302, 211-293(1998)