Finding Zeros of Nonlinear Functions Using the Hybrid Parallel Cell Mapping Method

Fu-Rui Xiong[a] Oliver Schütze[b] Qian Ding[a] Jian-Qiao Sun[c]

[a]*Department of Mechanics*
Tianjin University, Tianjin, 300072, China

[b]*CINVESTAV-IPN, Depto de Computacion*
Mexico City, 07360 Mexico

[c]*School of Engineering, University of California*
Merced, CA 95343, USA
Corresponding author and Honorary Professor of Tianjin University.
(jqsun@ucmerced.edu)

Abstract

Analysis of nonlinear dynamical systems including finding equilibrium states and stability boundaries often leads to a problem of finding zeros of vector functions. However, finding all the zeros of a set of vector functions in the domain of interest is quite a challenging task. This paper proposes a zero finding algorithm that combines the cell mapping methods and the subdivision techniques. Both the simple cell mapping (SCM) and generalized cell mapping (GCM) methods are used to identify a covering set of zeros. The subdivision technique is applied to enhance the solution resolution. The parallel implementation of the proposed method is discussed extensively. Several examples are presented to demonstrate the application and effectiveness of the proposed method. We then extend the study of finding zeros to the problem of finding stability boundaries of potential fields. Examples of two and three dimensional potential fields are studied. In addition to the effectiveness in finding the stability boundaries, the proposed method can handle several millions of cells in just a few seconds with the help of parallel computing in graphics processing units (GPUs).

Key words: Zeros Finding; Simple Cell Mapping; Generalized Cell Mapping; Parallel Computing; Subdivision; Stability Boundary.

1 Introduction

Finding zeros of multi-variable nonlinear functions is a common problem existing in many scientific and engineering fields. In the area of dynamics, finding equilibrium states of nonlinear systems, bifurcation and stability analysis of the system all lead to zero finding of nonlinear functions. In control systems, the stability region in the parameter space can also be transformed to a zero finding problem. This paper presents an algorithm using the simple cell mapping and generalized cell mapping that can find zeros of multi-variable nonlinear functions in an efficient manner.

Since analytical solutions for zeros of nonlinear functions are in general difficult to obtain, there have been many studies of numerical methods for zero finding. The classical Newton's method with gradient information has been successfully applied to various problems [1]. A number of novel variations of Newton's method are popular choices for many applications [2, 3]. Other algorithms are focused on the non-smooth or complex functions where the derivatives are not available [4].

Other studies have been carried out to find global solutions of the zero finding problem in certain domain by treating either gradient based or gradient free search algorithms as point-mappings in the parameter space, and studying the long term evolutionary of the point-mappings. The homotopy continuation method [5], cell mapping [2] and set-oriented method [6, 7] have been applied by many scholars to search global solutions. The homotopy continuation method is performed in the continuous parameter space while the latter two methods are performed in the discretized parameter space. For problems with moderate to high dimensions, the point-wise methods become less feasible due to the increased need of computational efforts. The set-oriented method and its predecessor, the cell mapping method, are more computationally effective.

Either gradient based or gradient free point-to-point iterative search algorithm for finding zeros forms a point-mapping dynamical system in the parameter space. Hence, finding function zeros can be equivalently treated as finding global invariant sets of the point-mapping dynamical system. Both the cell mapping and set-oriented methods were originally developed for finding global invariant sets of nonlinear dynamical systems. The cell mapping technique was originally developed by Hsu in the 1980s [8] for global analysis of strongly nonlinear dynamical systems. One prominent feature of the cell mapping is that it stores only short term trajectories of the dynamical system in the discrete cell state space. The cell mapping is constructed by converting point-to-point mapping to cell-to-cell mapping. Each cell is designated by its central point and is assigned with an n-tuple integer coordinate where n is the dimension of the cell state space. An unravelling algorithm has been developed to extract global information from the cell mapping [8]. Two types of cell mappings, namely the simple cell mapping (SCM) and generalized cell mapping (GCM), have been developed. The simple cell mapping accepts only one image cell of a given cell, while the generalized cell mapping allows multiple image cells. The cell mapping method has been successfully applied in many fields (See references in [9]).

A further development of the cell mapping with an addition of subdivision technique, known as the set-oriented method, has been introduced by Dellnitz and Hohmann [10]. The set-oriented method with subdivision integrates an adaptive refinement technique. The advantage of the set-oriented method is that it can adaptively enhance the solution resolution with relatively lower computational effort by focusing on the invariant set only. This is known as the subdivision and selection procedure [11]. There have been a number of studies with the set-oriented method on searching invariant set and unstable manifolds. An adaptive subdivision algorithm was developed in [11] that allows the coexistence of multiple cell sizes to cover the invariant set. A study of global attractors of non-smooth mechanical system was carried out by Neumann *et al.* using the standard set-oriented method [12]. The set-oriented method is also a powerful tool in designing optimal controls [13, 14], especially for multi-objective optimal controls [15,16]. Furthermore, the set-oriented method has shown great performance with the capability of locating all solutions in both real and complex domains [6, 7].

This paper presents an algorithm of finding zeros of nonlinear functions in real domain. A SCM-GCM hybrid method in conjunction with the subdivision technique is developed to find zeros of nonlinear functions. For zero finding problems in high dimensional space, major components of the SCM-GCM hybrid method are parallelized. The parallel computing is carried out in a graphics processing unit (GPU) based upon the CUDA architecture. As a variant of zero finding studies, a searching algorithm for stability boundaries of nonlinear dynamical systems is proposed. The boundary searching algorithm makes use of the parallel simple cell mapping.

The reminder of this article is organized as follows. Section 2 states the zero finding problem. Section 3 reviews the cell mapping methods, namely, the simple cell mapping and generalized cell mapping. Section 4 presents the detail of the algorithm and discusses the parallelization of the SCM-GCM hybrid method. Section 5 validates the method with several test examples. An analysis is carried out to study the convergence of the solutions in the cell space. The algorithm for searching stability boundaries with the parallel simple cell mapping is presented and validated in Section 6. We close the paper with concluding remarks in Section 7.

2 Zero Finding Algorithm as Point Mapping

Consider the problem of finding zeros in the following

$$\begin{gathered}\mathbf{f}(\mathbf{x}) = 0, \\ \mathbf{f} : \mathbf{R}^m \to \mathbf{R}^n,\ \mathbf{x} \in \mathbf{U} \subset \mathbf{R}^m,\end{gathered} \tag{1}$$

where $\mathbf{U}$ is in a bounded region in $\mathbf{R}^m$. As an example, we consider the damped Newton's method for local search of zeros,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \gamma \mathbf{J}^\dagger(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k) \equiv \mathbf{F}(\mathbf{x}_k), \tag{2}$$

where $\mathbf{J}^\dagger$ denotes the pseudo inverse of the Jacobian matrix $\mathbf{J}$ of the function $\mathbf{f}(\mathbf{x}_k)$ at $\mathbf{x}_k$, and γ is a step size. Note that the algorithm allows both under-determined $(m > n)$ and over-determined $(m < n)$ cases. When no explicit expression of the function $\mathbf{f}(\mathbf{x})$ is available, the Jacobian matrix $\mathbf{J}$ must be numerically computed with the methods such as the finite difference.

We are not interested in the case when the system has one simple zero in the region $\mathbf{U}$. Our goal is to find all the zeros of the vector-valued function $\mathbf{f}(\mathbf{x})$ for $\mathbf{x} \in \mathbf{U}$. In particular, we would like to find the zeros that form the set with certain global structure. Such zero solutions would require prohibitively extensive point-wise searches. However, the cell mapping technique is extremely attractive for this kind of problems.

Equation (2) defines a point-to-point mapping $\mathbf{F} : \mathbf{R}^m \to \mathbf{R}^m$. We introduce the concept of the global attractor of the mapping $\mathbf{F}$. Define a set A such that

$$\mathbf{F}(A) = A, \quad A \in \mathbf{U}. \tag{3}$$

Set A is called the global attractor in $\mathbf{U}$ if $\bigcup\limits_{i=1}^{\infty} \mathbf{F}^{-i}(A_\varepsilon) = \mathbf{U} \subset \mathbf{R}^m$, where A_ε is a set covering A and its ε-neighborhood. By definition, for any $\mathbf{z} \in A$, $\mathbf{F}^i(\mathbf{z}) \in A$ for $\forall i \in N^+$. N^+ denotes the set of the positive integers. Moreover, $\bigcap\limits_{i=1}^{\infty} \mathbf{F}^i(B) \subseteq A$ holds if $B \subseteq A$.

Finding zero is equivalent to solving an unconstrained minimization problem as

$$\min_{\mathbf{x} \in \mathbf{U}} \|\mathbf{f}(\mathbf{x})\|_2 . \tag{4}$$

The Armijo rule, also known as the linear search algorithm, to determine the step length γ for searching the minimum of Equation (4) in the continuous space [17]. In the cell space, the smallest step length γ is determined such that $\left\|\gamma \mathbf{J}^\dagger(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)\right\|_2$ is smaller than the half of the minimum cell dimension. This is because when the one step point-to-point mapping from the center of a cell lands within the same cell, this cell must be an attractor and a candidate of the solution.

3 Cell Mapping

3.1 Simple Cell Mapping

Recall that Equation (2) defines a nonlinear point-to-point mapping $\mathbf{F} : \mathbf{R}^m \to \mathbf{R}^m$. The cell mapping methods propose to discretize the domain $\mathbf{U} \subset \mathbf{R}^m$ into a discrete space $\mathbf{Z}^m$ consisting of small cells. The point-to-point mapping is replaced with the cell-to-cell mapping. The simple cell mapping starts from the center of a cell, computes the mapping of the center from Equation (2), and identifies the cell that contains the image point as the image cell. When the domain $\mathbf{U}$ is finite, the number of cells in $\mathbf{Z}^m$ is also finite so that each cell can be indexed by one integer denoted as z. Let Z denote the set of integers representing the cells in $\mathbf{Z}^m$. The simple cell mapping in $\mathbf{Z}^m$ is symbolically denoted as

$$z_{k+1} = C(z_k), \quad z_k \in Z, \tag{5}$$

where the subscript k indicates the k^{th} step of the mapping, corresponding to the k^{th} search step in zero finding problems.

An unravelling algorithm can be used to analyze the path generated from each cell and assign the group Gr, step St and periodicity Pe numbers properly [8]. Once all the cells in Z are assigned with these numbers, the global invariant solutions can be discovered from the SCMs.

3.2 Generalized Cell Mapping

Since the SCM only considers the center of a cell, the transformation of the point-to-point mapping to the SCM will inevitably bring in errors. These errors exist in the one step cell mappings and accumulate when the cell mappings are iterated. The long-term cumulative error can lead to false limit cycles or artificial attractors [18]. One way to tackle this problem is to sample more points in a cell to represent the cell in the construction of one step cell mappings. This leads to the concept of generalized cell mapping. A strict proof of sampling point technique for generating the generalized cell mapping has been provided in [19].

The GCM lands itself a probabilistic description of dynamic systems and the dynamics of the finite number of cells form a Markov chain. The evolution of the probabilities of the system is governed by the transitional probability equation $\mathbf{p}_{k+1} = \mathbf{P}\mathbf{p}_k$ where $\mathbf{p}_k$ is the probability distribution in the cell space at the k^{th} iteration and $\mathbf{P}$ the constant one-step transitional probability matrix of the Markov chain. Much of the well-established theory on Markov chains becomes readily applicable to the analysis of generalized cell mapping [20].

The one-step transitional probability matrix $\mathbf{P}$ can be viewed as a directed graph, and can be stored in the form of sparse matrix. In the analysis of dynamic systems, the logical matrix or topological matrix denoted as $\mathbf{G} = \{g_{ij}\}$ of $\mathbf{P}$ is of particular importance. The matrix $\mathbf{G}$ is defined such that when z_i has z_j as one of its images, $g_{ij} = 1$, otherwise, $g_{ij} = 0$. Clearly, $\mathbf{G}^T$ is the topological matrix of the inverse dynamics of $\mathbf{G}$.

The global invariant solutions of the system such as attractors manifest as cyclic or irreducible sets of cells in the GCM. The sets of invariant cells are called as the *persistent groups* (PGs). A PG is the set such that the images of all cells still belong to the set. Clearly, a PG is an attractor in the cell space [20, 21].

Although the results obtained with the GCM contain much richer information about the global properties of nonlinear dynamic systems, it is more computationally intensive than the SCM. Therefore, combining the GCM with SCM in a strategic manner may offer a balance of computational effort and accuracy of the solution. This leads to the SCM-GCM hybrid algorithm.

4 Parallel Hybrid Algorithm

The general idea of this algorithm is to obtain the coarse covering set of zeros first and then conduct subdivision and selection based on the coarse set until the results are well refined. The coarse covering set is acquired by a hybrid implementation of SCM and GCM. The selection of sub-divided cells makes use of both the backward GCM and forward SCM. The algorithm is outlined as follows:

1. Discretize the search space coarsely to form the first cell space S_1. Construct the topological matrix $\mathbf{G}$ of the GCM in S_1 with properly distributed sample points in each cell.

2. Select a SCM from the GCM, which is treated as a collection of the SCMs, and identify the periodic cells.

3. For each periodic cell z_i, search the set P_i that z_i leads to and the set Q_i that leads to cell z_i. The covering set S_c is expressed as

$$S_c = \bigcup_{i=1}^{N_m} \left[(P_i \cap Q_i) \cup z_i \right], \tag{6}$$

where N_m is the total number of periodic cells.

4. Refine S_c by sub-dividing the cells in it, and denote the refined set as S_{cr}. Build the GCMs in S_{cr} and record the topological matrix $\mathbf{G}$. Select the backward invariant cell set S_r as,

$$S_r = \left\{ z \mid \mathbf{G}^{-1}(z) \cap S_{cr} \neq \emptyset \right\}. \tag{7}$$

5. Replace S_c with S_r. Execute steps 4 and 5 for N_{GCM} times.

6. Refine S_r by sub-dividing the cells in it. Build the SCMs in S_r. Find the periodic cells in the SCMs. Let S_r contain the periodic cells.

7. Refine the periodic cells with sub-division. Repeat steps 6 and 7 for N_{SCM} times.

8. If needed, conduct post-processing by applying point-wise searches starting from the centers of the cells of the final SCM solution.

Some remarks on the algorithm are in order. When the cell space is coarse, a sufficiently large number of points in a cell must be sampled to generate multiple image cells for the GCM. The selected SCM in Step 2 has the maximum transitional probability among other mappings in the GCM, which has been introduced in Hsu's book [20]. If more than one image with the maximum transitional probability is found, we randomly select one SCM from them. Note that the backward invariant set of cells obtained from Equation (7) contains the unstable equilibrium states of the underlying dynamic system [12, 19]. In this study, such a property is exploited to recover the covering set completely at the first few iterations of sub-division when the cell space is coarse.

To implement Equation (6), one can use the *depth first search* (DFS) algorithm in the graph theory to obtain P_i and Q_i by traversing the GCM graph from z_i forward and backward [22]. Recall that the topological matrix of the inverse GCM $\mathbf{G}^{-1}$ is the same as its transpose. Alternatively, S_c can be gradually built by expanding the set started from the periodic cells. In doing so, one needs to bring in all image cells of the expanding S_c until it cannot be expanded anymore. This approach generates a PG set that has each of its element never mapped out of the set.

The sampling point method to construct GCMs for low dimensional problems can use uniform sampling in a cell. However, for high dimensional problems, the uniform sampling quickly becomes impractical due to the exponential growth of the number of points. In this work, we propose random sampling for high dimensional problems.

In summary, the hybrid SCM-GCM algorithm utilizes the invariant set searching capability for both the cell mapping methods. The GCM method is used to handle coarse cells while the SCM approach is applied to the small cells with sufficient accuracy.

4.1 Parallel Implementation

Most computations of the SCM and GCM method are highly parallelizable. The construction of the GCM, search for periodic cells of the SCM and the GCM based backward search consume the majority of execution time. Parallelization of these can speed up the computations of the GCM-SCM hybrid method significantly. In this work, the parallel computations make use of the CUDA architecture for the NVIDIA graphics processing cards. The CUDA architecture treats the cores in the graphics processing card as independent computing units and performs the data parallelization.

Let T denote as the total number of threads allocated from the GPU, T_{id} as the current identity of the thread under processing, c_{id} as the cell number that is processed on thread T_{id}. In parallel computing, multiple threads are executed simultaneously. On thread T_{id}, only cells with cell number $c_{id} + kT$ are processed, where k is a positive integer. Hence, in the loop to sweep all the cells, each step processes T cells in the parallel computing.

Tables 1-4 list the pseudo code for creating cell mappings, searching for periodic cells in SCMs, and finding the covering set with the backward search in GCMs. Except for finding the covering set, which uses the sequential DFS algorithm, all other components are parallelizable. It should be noted that in parallel computing, allocating memory in a dynamic manner will largely decrease the overall performance. It is therefore highly recommended to fix the array size and use an associated flag array to determine which cell to keep in the next iteration. The computations are conducted on the GPU while the data collection is on the CPU. One should be aware that the data transmission between GPU and CPU is an overhead limiting the speed up of the code.

Table 5 presents the main structure of the parallel SCM-GCM algorithm for zero finding. The only sequential part of the algorithm is at line 4 for finding the covering set. There are several user defined parameters as input of the algorithm: the maximum period $p_{\max}$ for searching the periodic cells in SCM, initial cell space partition N, iterations N_{GCM} of the GCM and iterations N_{SCM} of the SCM. The periods of the groups of cells in the SCM are less than or equal to $p_{\max}$. We have found that $p_{\max} = 20$ can ensure all cyclic structures to be captured. The selection of N_{GCM} and N_{SCM} is a trade-off between accuracy and computational effort. A large N_{GCM} means the covering set closer to be complete in the refined cell space and involves more cells. On the other hand, a small N_{GCM} may lead to an incomplete covering set when the cells are coarse due to inadequate sampling points or local mapping error, resulting in false periodic cells or even no periodic cells [23]. Furthermore, once a big cell is missed during the search for periodic cells, the SCM method will not recover that region since there is no available mapping being created in it.

4.2 Post-Processing

Through extensive numerical experiments, we find that the proposed hybrid cell mapping algorithm always returns the periodic cells with period one, also known as the $P-1$ cells, at the final stage. This is due to the descending nature of the underlying dynamics generated by the zero search algorithm. Therefore, the SCM tends to converge to solutions with zero gradient, namely, stationary states. This property is different from the SCM method applied to nonlinear dynamical systems governed by differential equations, where the SCMs are derived from the short-time integration of the differential equations.

Since the true point-wise solution of zeros must lie in the $P-1$ cells of the SCMs, the centers of the $P-1$ cells can be taken as approximate solutions of the zeros. The error of the solutions is of the order $O(h/2)$, where h is the largest dimension of the cells in the final refined cell space. For low dimensional problems, a sufficiently small h can be used through multiple subdivisions, thus leading to highly accurate solutions. For high dimensional problems, a very small h would imply the need for exceedingly large memory and CPU time. The post-processing of the cell mapping solutions for high dimensional problem is usually needed.

The original goal of the cell mapping method is not to acquire highly accurate point-wise solutions. Instead, the cell mapping results provide good initial conditions and database for further point-wise solutions if needed. An example of such a study is the interpolated cell mapping method [24–26], which practically constitutes a post-processing of the cell mapping results. The complete zero finding solution method proposed here consists of two steps from cell level approximation to point-wise solutions. This solution process has two advantages that most contemporary zero finding algorithms do not enjoy: a) the point-wise searching starting from the centers of $P-1$ cells converges fast since these cells contain the true solutions; b) the set of the point-wise solutions is more likely global and complete in the sense that all the zeros are found in the domain which has been swept by the proposed hybrid cell mapping method. On the other hand, if only the point-wise search is used to find all the zeros in a domain of the high dimensional space, the computational effort would be prohibitive.

5 Numerical Examples

The computations of all examples reported in this paper are conducted on a Lenovo laptop equipped with the NVIDIA GeForce GT 755M graphics card. This card has 384 CUDA cores. Its clock is 980 MHz. The programming is under the CUDA C/C++ environment.

5.1 $\mathbf{R}^2 \rightarrow \mathbf{R}^2$ *Vector Functions*

We first consider a 2×2 vector function [6].

$$\mathbf{f}(x_1, x_2) = \begin{bmatrix} 4x_1(x_1^2 + x_2 - 11) + 2(x_1 + x_2^2 - 7) \\ 2(x_1^2 + x_2 - 11) + 4x_2(x_1 + x_2^2 - 7) \end{bmatrix} \tag{8}$$

This function is known to have 9 zeros. We initially partition the region of interest $[-5, 5] \times [-5, 5]$ into a 13×13 cell space. 3×3 points are uniformly sampled in a cell to construct the GCMs. One edge at a time of the cell is divided by 3 in the sub-divisions of the cell space. For this two dimensional problem, we first divide the cell along x_1 direction by 3 and then along x_2 by 3. By doing so, the increase of number of cells can be generally controlled at a linear rate. This approach, known as the *rolling sub-division*, was originally designed for high dimensional problems.

The covering set and clusters of the final refined cells representing the zeros obtained by the SCM-GCM method are depicted in Figure 1. For this simple problem, we do not need GCM and have taken $N_{GCM} = 0$ and $N_{SCM} = 8$. 28 coarse cells are found as the covering set of zeros. 91 periodic cells with period one in the 1053×1053 refined cell space are found to form 9 clusters representing 9 zeros of the function. The computation takes 1.20 seconds. Post-processing is conducted with the cell mapping solutions as the initial guess for the point-wise algorithm. Within the order of 10^{-12}, we have found 9 zeros which are listed in Table 6. The post-processing takes 0.6864 seconds.

We emphasize that the results shown in Table 6 cannot be efficiently acquired by the point-wise search without sweeping over a large number of initial points. In addition, there is no guarantee that the point-wise search can find all the solutions of the nonlinear equations in a reasonable time frame.

Figure 2 shows the convergence of the solution as a function of subdivisions. The error is bounded by $\max(h/2)$, where h denotes the largest cell size in the final subdivision. For lower dimensional problems, we can afford to have very small cells so that the zero solutions obtained by the SCM-GCM method and represented by the centers of the periodic cells can be sufficiently accurate. For higher dimensional problems, the final cell size can still be large. Post-processing becomes necessary.

5.2 $\mathbf{R}^6 \rightarrow \mathbf{R}^6$ *Vector Functions*

Next, consider a higher dimensional problem with 6 variables. This is a neuroscience model discussed in [27]. The nonlinear equations are,

$$\begin{cases} x_1^2 + x_3^2 - 1 = 0 \\ x_2^2 + x_4^2 - 1 = 0 \\ x_5 x_3^3 + x_6 x_4^3 = 0 \\ x_5 x_1^5 + x_6 x_2^3 = 0 \\ x_5 x_1 x_3^2 + x_6 x_4^2 x_2 = 0 \\ x_5 x_1^2 x_3 + x_6 x_4 x_2^2 = 0 \end{cases} \tag{9}$$

We search the zeros in the region $x_i \in [-1, 1]$ $(i = 1, 2, \cdots, 6)$. The initial partition of the search region is set to be 2 along all six direction. The rolling sub-division technique is used with each direction divided by 3. In total 18 iterations of sub-divisions are carried out leading to the final cell space such that the region $[-1, 1]$ for each x_i is divided by 54. The number of the GCM backward search is selected to be $N_{GCM} = 2$. The number of the SCM forward searches is set to be $N_{SCM} = 16$. 80 randomly sampled points from each cell are used to construct the GCMs. 38 covering cells are found in the first run. 11176 refined $P-1$ cells are found. The periodic cells form a large number of clusters representing the solutions of Equation (9). The parallel SCM-GCM computing takes 38.5951 seconds. Post-processing takes 16.5347 seconds. The point-wise search in the post-processing uses `fsolve` function in Matlab with parallel computing toolbox to speed up the solving process. The cells representing the solutions only occupy a small percentage of the cell space ($4.5074 \times 10^{-5}\%$).

In Figure 3, blue dots in the left figure are the centers of the cells and red dots in the right figure are the converged point-wise solutions. The solutions form complicated geometric structures. The cell mapping method discovers the global and complex structures of the solution, and provides good initial conditions for the point-wise search algorithms to obtain the true solutions far more efficiently. The point-wise solutions depicted in the right figure all have the accuracy of order 10^{-9}. To our best knowledge, no studies of global solutions of Equation (9) have been reported in the literature.

5.3 $\mathbf{R}^{10} \rightarrow \mathbf{R}^{10}$ *Vector Functions*

As the final example of finding zeros, we compute the equilibrium states of an economic model,

$$\begin{cases} \left(x_k + \sum\limits_{i=1}^{n-k-1} x_i x_{i+k} \right) x_n = 0 \\ \sum\limits_{j=1}^{n-1} x_j + 1 = 0 \end{cases}, 1 \leq k \leq n-1 \tag{10}$$

This equation was studied by applying the multi-objective optimization technique reported in [27]. In the present study, we take $n = 10$. The search region $x_i \in [-0.8, 0.8]$ $(0 \leq i \leq 10)$ is divided by 2 along each axis x_i, leading to 2^{10} cells initially. We take $N_{GCM} = 2$ and $N_{SCM} = 14$ for GCM and SCM iterations. 300 points are randomly sampled in each cell to construct the GCMs. After 16 iterations of subdivisions, 759 $P-1$ cells are found to represent the solution of zeros. The total computing time is 258.041 seconds. 10 selected solutions obtained after the post-processing with the point-wise search algorithm are listed in Table 7. The post-processing takes 1.6044 seconds with the same computational resources reported in the previous example. It should be pointed out that the solutions listed in Table 7 are within the same cells. from the center of which the point-wise search starts. This implies that the cell mapping solutions contain the true solutions. The post-processing can converge to the true solution within the same cell in just a few iterations.

6 Stability Boundary of Potential Fields

6.1 Mathematical Background

In this section, we apply the cell mapping based zero finding algorithm to compute the stability boundary of potential fields [28–30]. Let $V(\mathbf{x})$ be a potential field of a nonlinear system where $\mathbf{x} \in \mathbf{R}^n$. The local minima of $V(\mathbf{x})$ can be considered as the stable equilibria of the dynamical system

$$\frac{d\mathbf{x}}{dt} = g(\mathbf{x}) = -\nabla V. \tag{11}$$

The equilibria of the dynamical system (11) include sinks, saddles and sources of the potential field. Finding all the equilibria of the potential field is the same as finding zeros of ∇V. The stability boundaries of stable equilibria are the union of stable manifold of saddles [30]. The definition of stable manifold of an equilibrium $\mathbf{x}_0$ is as follows,

$$W^s(\mathbf{x}_0) = \left\{ \mathbf{x} \in \mathbf{R}^n : g^k(\mathbf{x}) \to \mathbf{x}_0 \text{ as } k \to \infty \right\}. \tag{12}$$

Let $\mathbf{x}_i$ be a saddle of the dynamical system (11), more specifically, type-1 saddle with only one of the eigenvalues of the local Jacobian matrix having positive real part. Reference [30] proved that the stability boundaries ∂A are given by

$$\partial A = \bigcup_i W^s(\mathbf{x}_i). \tag{13}$$

Next, we apply the SCM-GCM method to compute ∂A. Recall that the domains of attraction of the stable equilibria delineate the stability boundaries [29]. The SCM-GCM method is first applied to find local minima and maxima of $V(\mathbf{x})$, i.e. the stable and unstable equilibrium points, and then to outline their domains of attraction of the stable equilibria. The latter task is what the cell mapping methods were invented for.

6.2 Stability Boundary Computing Algorithm

The stability boundary computing consists three basic parts: 1) finding zeros of $\nabla V = \mathbf{0}$ as equilibrium points; 2) stability analysis of equilibrium points; 3) finding the domains of attraction. The local stability analysis of equilibrium points is determined by the eigenvalues of the Jacobian matrix of Equation (11).

In this study, the cell mapping methods must be applied to the entire cell space. It is also difficult to apply subdivisions.

When computing cell mapping for a cell z, the potential energy $V(\mathbf{x})$ is evaluated at the center of z, and is denoted as $V(z)$. We pick cell z_{img} among the neighbor cells of z whose energy value is lower than $V(z)$ and has the biggest decrease. If such a z_{img} can be found, we assign it as the image cell of z.

When the image cell cannot be found around the neighbors of z, we apply the Nelder–Mead simplex method [31] to find the "true" local minimum of $V(\mathbf{x})$ with the center of cell z as the initial guess. The image cell of z is taken as the cell where the simplex method converges to when looking for the minimum of $V(\mathbf{x})$. It should be noticed that the point-wise searching of image cell is only applied when no feasible image cell can be found by comparing among the neighbor cells. For most of the transient cells, the neighborhood based image cell search is effective and fast. The pseudo code of this algorithm is listed in Table 8.

The unraveling algorithm proposed by Hsu is a sequential procedure that can categorize each cell by its attraction domain [20]. The power of this unraveling algorithm is that it also unveils the behavior of transient cells. In this study, we only need to categorize the cells by the group number Gr. The algorithm is presented in Table 9. A parallel version of this algorithm is shown in Table 2, which detects the cyclic structure only. The sequential and parallel unraveling algorithms extract different features from the SCM. The unraveling algorithm by Hsu assigns a group number to each cell indicating to which equilibrium the cell is attracted. The stability boundaries consist of the cells whose neighbors have different group numbers. The algorithm to extract the boundary defined in Equation (13) is shown in Table 10. The search starts from saddle cells, denoted as TAR in Table 10). The logical flow of the stability boundary search algorithm is listed in Table 11.

6.3 Test Examples

6.3.1 2D Muller–Brown Potential

The two dimensional Muller-Brown potential for characterizing a chemical process [32] is commonly used to test algorithms to find saddle points.

$$V(x_1, x_2) = \sum_{i=1}^{4} A_i \exp\left[a_i(x_1 - x_i^0)^2 + b_i(x_1 - x_i^0)(x_2 - y_i^0) + c_i(x_2 - y_i^0)^2\right], \tag{14}$$

where

$$\begin{aligned} A &= [-200, -100, -170, -15]\,, & a &= [-1, -1, -6.5, -0.7]\,, \\ x^0 &= [1, 0, -0.5, 1]\,, & b &= [0, 0, 11, 0.6]\,, \\ y^0 &= [0, 0.5, 1.5, 1]\,, & c &= [-10, -10, -6.5, 0.7]\,. \end{aligned} \tag{15}$$

The gradient of the Muller-Brown potential reads,

$$\begin{aligned} \frac{\partial V}{\partial x_1} &= \sum_{i=1}^{4} A_i P\left[2a_i(x_1 - x_i^0) + b_i(x_1 - x_i^0)\right], \\ \frac{\partial V}{\partial x_2} &= \sum_{i=1}^{4} A_i P\left[b_i(x_1 - x_i^0) + 2c_i(x_2 - y_i^0)\right], \end{aligned} \tag{16}$$

where

$$P = \exp\left[a_i(x_1 - x_i^0)^2 + b_i(x_1 - x_i^0)(x_2 - y_i^0) + c_i(x_2 - y_i^0)^2\right]. \tag{17}$$

The search domain for stability boundaries is $[-1.5, 1.5] \times [-0.5, 2]$. The initial cell space partition is 7×7. The final cell space resolution is 567×567. 6 equilibrium points are found in this domain. Three stable sinks and three saddles are identified by the local stability analysis with help of the eigenvalues of the Jacobian matrix of the linearized potential function at the equilibriums. The results are shown in Table 12. The zero finding algorithm takes 1.4901 seconds.

Figure 4 presents the equilibria and stability boundaries when the cell space resolution is 135×135. Different colors of cells represent different groups. The white area in the lower left of the figure contains cells that are mapped out of the region of interest known as the sink cell. The stability boundaries, as outlined by the regions with different colors, consist of 907 cells. The boundary search of this example takes only 0.04296 seconds.

6.3.2 3D Lennard–Jones Potential

We now consider a three dimensional example of the Lennard–Jones potential. This potential is used to examine a saddle point search algorithm in [29].

$$\begin{aligned} V &= \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} v(r_{ij}), \\ v(r_{ij}) &= \epsilon \left[\left(\tfrac{r_0}{r_{ij}}\right)^{12} - 2\left(\tfrac{r_0}{r_{ij}}\right)^{6} \right], \end{aligned} \tag{18}$$

where

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}. \tag{19}$$

We set $\epsilon, r_0 = 1$, and $N = 3$ in this study. We further assume that the three atoms are constrained such that there are only three independent variables: (x_2, x_3, y_3). Other coordinates are set as zeros. The choice of the independent variables is somewhat arbitrary. The purpose of this example is to demonstrate the method. Although more independent variables can be readily handled with the proposed method, the visualization of the results is difficult. The gradient of Equation (18) reads,

$$\begin{aligned} \frac{\partial V}{\partial x_i} &= \sum_{j=1, j\neq i}^{3} \frac{12}{r_{ij}^8} \left[1 - \left(\frac{1}{r_{ij}}\right)^6 \right] (x_i - x_j), \\ \frac{\partial V}{\partial y_i} &= \sum_{j=1, j\neq i}^{3} \frac{12}{r_{ij}^8} \left[1 - \left(\frac{1}{r_{ij}}\right)^6 \right] (y_i - y_j), \end{aligned} \tag{20}$$

For this special case, where only (x_2, x_3, y_3) are the nonzero variables, Equation (20) reads,

$$\begin{aligned}
\frac{\partial V}{\partial x_2} &= \frac{12}{r_{21}^8}\left[1-\left(\frac{1}{r_{21}}\right)^6\right]x_2 + \frac{12}{r_{23}^8}\left[1-\left(\frac{1}{r_{23}}\right)^6\right](x_2-x_3), \\
\frac{\partial V}{\partial x_3} &= \frac{12}{r_{31}^8}\left[1-\left(\frac{1}{r_{31}}\right)^6\right]x_3 + \frac{12}{r_{32}^8}\left[1-\left(\frac{1}{r_{32}}\right)^6\right](x_3-x_2), \\
\frac{\partial V}{\partial y_3} &= \frac{12}{r_{31}^8}\left[1-\left(\frac{1}{r_{31}}\right)^6\right]y_3 + \frac{12}{r_{32}^8}\left[1-\left(\frac{1}{r_{32}}\right)^6\right]y_3,
\end{aligned} \tag{21}$$

where

$$r_{21} = |x_2|\,,\ r_{32} = \sqrt{(x_3-x_2)^2+y_3^2},\ r_{31} = \sqrt{x_3^2+y_3^2}. \tag{22}$$

The zeros of Equation (21) are presented in Table 13. The initial cell space partition for zero finding is 11 × 11 × 11 and the final resolution reaches 297 × 297 × 297. There are 10 equilibrium points with 4 sinks and 6 saddles. The distribution of equilibria are anti-symmetric with respect to the $x_3 - y_3$ plane. The zero finding of Equation (21) takes 2.2667 seconds.

Figure 5 display the boundary surfaces from different perspectives. The cell space partition for the whole domain sweeping with the cell mapping method is 120 × 120 × 120. The boundary surfaces generally divide the $3D$ space into four parts. Each part has one stable equilibrium as the attractor. Due to the symmetry of the system, the $x_3 - y_3$ plane is a part of the boundaries. The boundary surfaces consist of 82221 cells. The total computational time for the stability boundary is 3.8146 seconds.

7 Concluding Remarks

This paper proposes a SCM-GCM hybrid method combined with the subdivision technique, which unifies finding zeros of nonlinear equations and discovery of stability boundaries of complex potential fields. For zero finding problems, we have found that all the periodic cells of the SCM are $P-1$, which suggests that the zeros of the vector function lie inside the $P-1$ cells. For low dimensional problems, we can afford to use very small cells so that the results in the cell space can be sufficiently accurate. For high dimensional problems where the cells may still be large, post-processing such as interpolation or point-wise search can be used to improve the accuracy of the solution. The SCM-GCM hybrid method is highly parallelizable, which makes it a viable choice for high dimensional problems. The computations for complicated boundary surfaces of the 3 dimensional example involve millions of cells and are completed in just a few seconds.

Acknowledgements

This work is supported by the Natural Science Foundation of China through the grants (11172197 and 11332008), by the Natural Science Foundation of Tianjin through a key-project grant, and by the UC MEXUS-CONACyT through the project “Cell-to-Cell Mapping for Global Multi-objective Optimization”. The first author Fu-Rui Xiong would like to thank the China Scholarship Council (CSC) for sponsoring his study in the United States of America.

References

[1] K. Madsen, A root-finding algorithm based on Newton’s method, BIT (Nordisk Tidskrift for Informationsbehandling) 13 (1) (1973) 71–75.

[2] R. Carniel, Quasi cell mapping approach to the global dynamical analysis of Newton’s root-finding algorithm, Applied Numerical Mathematics 15 (2) (1994) 133–152.

[3] A. Bhaya, E. Kaszkurewicz, Newton algorithms via control Liapunov functions for polynomial zero finding, in: Proceedings of the 43rd IEEE Conference on Decision and Control (CDC), Vol. 2, Piscataway, New Jersey, 2004, pp. 1629–1634.

[4] T. R. Chandrupatla, New hybrid quadratic/bisection algorithm for finding the zero of a nonlinear function without using derivatives, Advances in Engineering Software 28 (3) (1997) 145–149.

[5] Z.-J. Liu, An algorithm for finding all isolated zeros of polynomial systems, in: Proceedings of the International Symposium on Symbolic and Algebraic Computation, New York, USA, 1990.

[6] M. Dellnitz, O. Schütze, S. Sertl, Finding zeros by multilevel subdivision techniques, IMA Journal of Numerical Analysis 22 (2) (2002) 167–185.

[7] M. Dellnitz, O. Schütze, Z. Qinghua, Locating all the zeros of an analytic function in one complex variable, Journal of Computational and Applied Mathematics 138 (2) (2002) 325–333.

[8] C. S. Hsu, A theory of cell-to-cell mapping dynamical systems, Journal of Applied Mechanics 47 (1980) 931–939.

[9] J.-Q. Sun, L. Hong, Fuzzy cell mapping, in: J.-Q. Sun, A. C. J. Luo (Eds.), Global Analysis of Nonlinear Dynamics, Springer New York, 2012, pp. 161–174.

[10] M. Dellnitz, A. Hohmann, A subdivision algorithm for the computation of unstable manifolds and global attractors, Numerische Mathematik 75 (3) (1997) 293–317.

[11] M. Dellnitz, O. Junge, An adaptive subdivision technique for the approximation of attractors and invariant measures, Computing and Visualization in Science 1 (2) (1998) 63–68.

[12] N. Neumann, T. Sattel, J. Wallaschek, On set-oriented numerical methods for global analysis of non-smooth mechanical systems, Journal of Vibration and Control 13 (9-10) (2007) 1393–1405.

[13] O. Junge, H. M. Osinga, A set oriented approach to global optimal control, ESAIM: Control, Optimisation and Calculus of Variations 10 (2004) 259–70.

[14] L. Grune, H. Junge, A set oriented approach to optimal feedback stabilization, Systems & Control Letters 54 (2) (2005) 169–80.

[15] O. Schütze, K. Witting, S. Ober-Blöbaum, M. Dellnitz, Set oriented methods for the numerical treatment of multiobjective optimization problems, in: Proceedings of EVOLVE - A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation, Vol. 447, Leiden, The Netherland, 2013, pp. 187–219.

[16] M. Blesken, U. Ruckert, D. Steenken, K. Witting, M. Dellnitz, Multiobjective optimization for transistor sizing of CMOS logic standard cells using set-oriented numerical techniques, in: Proceedings of NORCHIP, Piscataway, New Jersey, 2009.

[17] L. Armijo, Minimization of functions having Lipschitz continuous first partial derivatives, Pacific Journal of Mathematics 16 (1) (1966) 1–3.

[18] L. G. Crespo, J. Q. Sun, Solution of fixed final state optimal control problems via simple cell mapping, Nonlinear Dynamics 23 (2000) 391–403.

[19] M. Dellnitz, O. Junge, Set oriented numerical methods for dynamical systems, Handbook of Dynamical Systems 2 (2002) 221–264.

[20] C. S. Hsu, Cell-to-Cell Mapping, Springer-Verlag, New York, 1987.

[21] L. Hong, J. Xu, Crises and chaotic transients studied by the generalized cell mapping digraph method, Physics Letters A 262 (4-5) (1999) 361–375.

[22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, The MIT Press, Cambridge, Massachusetts, 2009.

[23] L. G. Crespo, Deterministic and stochastic optimal control and cell mapping methods, Ph.D. thesis, University of Delaware (2002).

[24] B. H. Tongue, On obtaining global nonlinear system characteristics through interpolated cell mapping, Physica D: Nonlinear Phenomena 28 (3) (1987) 401–408.

[25] B. H. Tongue, K. Gu, Interpolated cell mapping of dynamical systems, Journal of Applied Mechanics 55 (1988) 461–466.

[26] B. H. Tongue, K. Gu, A theoretical basis for interpolated cell mapping, SIAM Journal on Applied Mathematics 48 (5) (1988) 1206–1214.

[27] C. Grosan, A. Abraham, A new approach for solving nonlinear equations systems, IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans 38 (3) (2008) 698–714.

[28] L. Luyckx, M. Loccufier, E. Noldus, Computational methods in nonlinear stability analysis: stability boundary calculations, Journal of Computational and Applied Mathematics 168 (12) (2004) 289–297.

[29] C. K. Reddy, H.-D. Chiang, A stability boundary based method for finding saddle points on potential energy surfaces, Journal of Computational Biology 13 (3) (2006) 745–766.

[30] H. D. Chiang, M. W. Hirsch, F. F. Wu, Stability regions of nonlinear autonomous dynamical systems, IEEE Transactions on Automatic Control 33 (1) (1988) 16–27.

[31] J. A. Nelder, R. Mead, A simplex method for function minimization, The computer journal 7 (4) (1965) 308–313.

[32] K. Müller, L. D. Brown, Location of saddle points and minimum energy paths by a constrained simplex optimization procedure, Theoretica Chimica Acta 53 (1) (1979) 75–93.

Table 1
Creation of generalized cell mapping and selected simple cell mapping.

```
Part 1: Cell mapping creation (parallel)
Input: Rough cell set $S$, point mapping $\mathbf{F}$, sampling number $N_s$
Output: GCM $\mathbf{G}$ and selected SCM $\mathbf{C}$
1: $T_{id} \leftarrow GetID()$
2: $c_{id} \leftarrow S_{T_{id}}$
3: while $c_{id} < totcells$
4:     for $i < N_s$
5:         $\mathbf{q} \leftarrow$random point within cell $c_{id}$
6:         $\mathbf{q}_i \leftarrow \mathbf{F}(\mathbf{q})$
7:         Record the cell where $\mathbf{q}_i$ belongs to into an array $R$
8:     end
9:     Store the unique elements in array $R$ as images of $c_{id}$
10:    Pick the cell in $R$ that has maximum transition probability
       and set $\mathbf{C}(c_{id})$ as that cell
11:    $c_{id} \leftarrow c_{id} + T$
12: end
```

Table 2
Finding the cyclic structure from simple cell mappings.

Part 2: Parallel SCM search

Input: Cell set S, SCM $\mathbf{C}$, maximum period $p_{\max}$

Output: Period of each cell P and seed cells S_e

```
1: $T_{id} \leftarrow GetID()$
2: $c_{id} \leftarrow S_{T_{id}}$
3: while $c_{id} < totcells$
4:     $c_{old} \leftarrow c_{id}$
5:     $flag \leftarrow false$
6:     for $i < p_{\max}$
7:         $c_{new} \leftarrow \mathbf{C}(c_{old})$
8:         if $c_{new} = c_{old}$ AND $c_{new}$ is not sink cell
9:             $P(c_{id}) \leftarrow i$
10:            $flag \leftarrow true$
11:            break
12:        end
13:        $c_{old} \leftarrow c_{new}$
14:    end
15:    if $flag = false$
16:        $P(c_{id}) \leftarrow -1$
17:    end
18:    $c_{id} \leftarrow c_{id} + T$
19: end
20: $Se$ is selected from the cells with $P \neq -1$
```

Table 3
Finding the covering set from seed cells.

```
**Part 3**: Find the covering set (sequential)
Input: Seed cells $S_e$, GCM $\mathbf{G}$
Output: Covering cells $S_c$ in coarse cell space
1: for $i < totcells$
2:     $P_i \leftarrow DFS(\mathbf{G}, Se_i)$
3:     $Q_i \leftarrow DFS(\mathbf{G}^{-1}, Se_i)$
4:     $i \leftarrow i + 1$
5: end
6: $S_c \leftarrow \bigcup_{i=1}^{totcells} [(P_i \cap Q_i) \cup Se_i]$
```

Table 4
Parallel backward searching.

```
Part 4: Parallel backward searching under GCM (parallel)
Input: Cell set $S$, GCM $\mathbf{G}$
Output: Invariant cells $S_r$
1: $T_{id} \leftarrow GetID()$
2: $c_{id} \leftarrow S_{T_{id}}$
3: while $c_{id} < totcells$
4:     if $\mathbf{G}^{-1}(c_{id}) \cap S = \emptyset$
5:         Mark $c_{id}$ as a non-invariant cell
6:     end
7:     $c_{id} \leftarrow c_{id} + T$
8: end
9: $S_r \leftarrow$Delete non-invariant cells from $S$
```

Table 5
The parallel SCM-GCM hybrid method for zero finding.

```
Parallel SCM-GCM algorithm for zero finding
Input: Searching bounds $lb, ub$, maximum period $p_{\max}$, cell space partition $N$
       refinement partition $sub$, underlying dynamics $\mathbf{F}$,
       GCM iteration time $N_{GCM}$, SCM iteration time $N_{SCM}$
Output: Invariant cell set $S_{inv}$ with central points as solutions
1: $iter \leftarrow 0$
2: $[\mathbf{G}, \mathbf{C}] \leftarrow$mapping creation($S, \mathbf{F}, N_s$)
3: $[P, S_e] \leftarrow$finding periodic cells($S, \mathbf{C}, p_{\max}$)
4: $S_c \leftarrow$covering set($S_e, \mathbf{G}$)
5: for $iter < N_{GCM}$
6:     $S_r \leftarrow$refine($S_c, N, sub$)
8:     Improve cell space resolution $N \leftarrow N \times sub$
9:     $[\mathbf{G}, \mathbf{C}] \leftarrow$mapping creation($S_r, \mathbf{F}, N_s$)
10:    $S_c \leftarrow$backward searching($S_r, \mathbf{G}$)
11:    $iter \leftarrow iter + 1$
12: end
13: $iter \leftarrow 0$
14: for $iter < N_{SCM}$
15:    $S_r \leftarrow$refine($S_c, N, sub$)
16:    Improve cell space resolution $N \leftarrow N \times sub$
17:    $\mathbf{C} \leftarrow$simple cell mapping creation($S_r, \mathbf{F}, N_s$)
18:    $[P, S_c] \leftarrow$ finding periodic cells($S_r, \mathbf{C}, p_{\max}$)
19:    $iter \leftarrow iter + 1$
20: end
21: $S_{inv} \leftarrow S_c$ and output the central points of $S_{inv}$ as approximations
```

Table 6
Solutions of Equation (8).

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$
-3.7793	-0.1280	3.5844	-0.2708	-3.0730	3.3852	3.0000	-2.8051	0.0867
-3.2832	-1.9537	-1.8481	-0.9230	-0.0814	0.0739	2.0000	3.1313	2.8843

Table 7
Selected solutions of Equation (10).

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$
−0.0297	−0.1266	−0.3460	0.2088	−0.7317
−0.7620	−0.0381	−0.7923	−0.0563	−0.7458
−0.0431	−0.7495	−0.3594	−0.4073	−0.4180
0.1330	−0.3991	0.4533	−0.2303	0.1141
0.5797	0.3978	0.4524	0.3948	0.4651
0.0541	0.3114	0.0517	−0.4943	−0.0495
−0.1362	−0.6633	0.3637	0.3932	0.1213
−0.1244	0.6644	−0.1566	−0.6741	0.4066
−0.6713	−0.3970	−0.6669	−0.1345	0.6511
−0.0000	−0.0000	−0.0000	−0.0000	−0.0000

$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$
−0.6826	−0.1875	−0.1304	−0.3059	−0.6104
−0.7868	−0.0373	−0.7513	−0.0414	−0.7513
0.3180	−0.1830	0.3111	0.3145	−0.4020
0.4917	−0.4821	0.4893	−0.7506	−0.5519
0.4742	0.4875	−0.0407	−0.3989	−0.0223
0.4784	−0.4624	0.0472	0.3102	0.0833
−0.4443	−0.4132	−0.3957	0.4007	0.6943
−0.4275	0.6739	−0.1315	−0.1306	0.1499
−0.4211	−0.3960	−0.3980	−0.3981	0.4105
−0.0000	−0.0000	−0.0000	−0.0000	0.0000

Table 8
Creation of simple cell mappings for the analysis of potential fields.

Program: Simple cell mapping creation of potential field analysis (parallel)

Input: Cell set S, potential field function V

Output: SCM $\mathbf{C}$

```
1: $T_{id} \leftarrow GetID()$
2: $c_{id} \leftarrow S_{T_{id}}$
3: while $c_{id} < totcells$
4:     $N \leftarrow$neighbor cells of $c_{id}$
5:     if $\exists z \in N$, $V(c_{id}) > V(z)$
6:         $\mathbf{C}(c_{id}) \leftarrow \arg\left\{\max_{z \in N}[V(c_{id}) - V(z)]\right\}$
7:     else
8:         $x_{img} \leftarrow$simplex algorithm$(V, x_{c_{id}})$
9:         $\mathbf{C}(c_{id}) \leftarrow$cell with $x_{img}$ locates in
10:    end
11:    $c_{id} \leftarrow c_{id} + T$
12: end
```

Table 9
Sequential unraveling algorithm of the simple cell mapping by Hsu.

```
Program: SCM unraveling (sequential)
Input: SCM $\mathbf{C}$, cell set $S$
Output: Group number array $Gr$
1: $Gr \leftarrow 0$, $g \leftarrow 1$
2: for $cs \in S$
3:     if $Gr(cs) = 0$
4:         $path \leftarrow \emptyset$
5:         while true
6:             $path \leftarrow path \cup cs$
7:             $cs\_new \leftarrow \mathbf{C}(cs)$
8:             $cs \leftarrow cs\_new$
9:             switch $Gr(cs)$
10:                case $0$
11:                    $Gr(cs) \leftarrow -1$
12:                    continue
13:                case $-1$
14:                    $g \leftarrow g + 1$
15:                    $Gr(path) \leftarrow g$
16:                    break
17:                otherwise
18:                    $Gr(path) \leftarrow Gr(cs)$
19:                    break
20:            end
21:        end
22:    end
23: end
```

Table 10
Boundary extraction from the results of the simple cell mapping analysis.

Program: Boundary extraction (sequential)

Input: Group number array Gr, saddle cells S_d, cell set S

Output: Boundary cell set M

```
1: $M \leftarrow \emptyset$, $BND\_OLD \leftarrow 0$, $BND\_NEW \leftarrow 0$, $TAR \leftarrow 0$
2: for $cs \in S_d$
3:     $TAR(cs) \leftarrow 1$, $BND\_NEW \leftarrow 1$
4:     while true
5:         for $ce \in S(TAR = 1)$
6:             $N \leftarrow$neighbor cells of $ce$
7:             for $cn \in N$
8:                 if $Gr(cn) \neq Gr(cs)$ AND $BND\_OLD(cn) = 0$
9:                     $TAR(cn) \leftarrow 1$, $BND\_NEW(cn) \leftarrow 1$
10:                end
11:            end
12:        $TAR(ce) \leftarrow 0$
13:    end
14:    if $\|BND\_OLD - BND\_NEW\| = 0$
15:        break
16:    else
17:        $BND\_OLD \leftarrow BND\_NEW$
18:    end
19: end
20: $M \leftarrow S(BND\_NEW = 1)$
```

Table 11
Main logic of the SCM method for computing stability boundaries.

SCM for computing the stability boundary of potential fields
Input: Potential field function $V(\mathbf{x})$, searching boundary lb, ub
Output: Boundary cell set M
1: P_Z $\leftarrow$SCM-GCM zero finding
2: Conduct local stability analysis of cell set P_Z and mark the saddle cells S_d
3: $\mathbf{C}$ $\leftarrow$SCM creation of potential field analysis
4: Gr $\leftarrow$Sequential algorithm for finding periodic cells
5: M $\leftarrow$Boundary extraction

Table 12
Equilibria of Equation (16).

Equilibria of the $2D$ potential field	
Saddles	Sinks
$(0.9782, 0.0084)$	$(-0.6770, -0.1572)$
$(0.0477, 0.4504)$	$(0.2296, 0.3639)$
$(-0.4868, 1.5194)$	$(-0.8464, 0.6021)$

Table 13
Equilibria of Equation (20).

Equilibria of the $3D$ potential field	
Saddles	Sinks
$(-1.0013, -2.0000, 0)$	$(-1, -0.5, -0.866)$
$(-0.9994, 0.9987, 0)$	$(1, 0.5, -0.866)$
$(-0.9994, -0.9987, 0)$	$(-1, -0.5, 0.866)$
$(-1.9987, -0.9994, 0)$	$(1, 0.5, 0.866)$
$(1.0013, 2.0000, 0)$	
$(1.9987, -0.9994, 0)$	

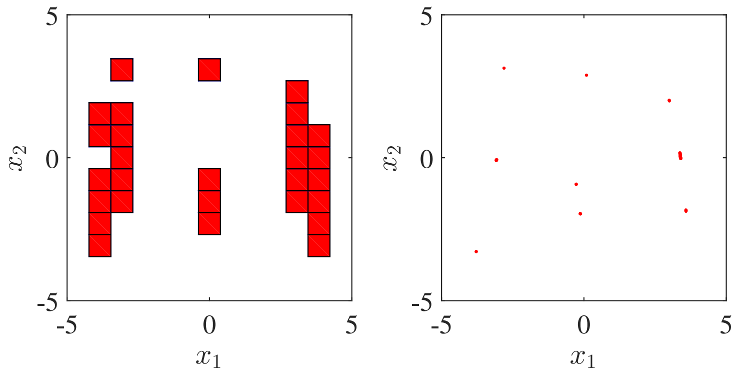

Fig. 1. Left: The covering set of zeros of Equation (8) with cell space partition 13×13. 28 cells are found. The covering set finding costs 0.69 seconds. Right: The zeros of Equation (8) after 8 subdivisions, represented by 9 clusters of 91 cells in the cell space with 1053×1053 partitions. Total computational time is 1.20 seconds.

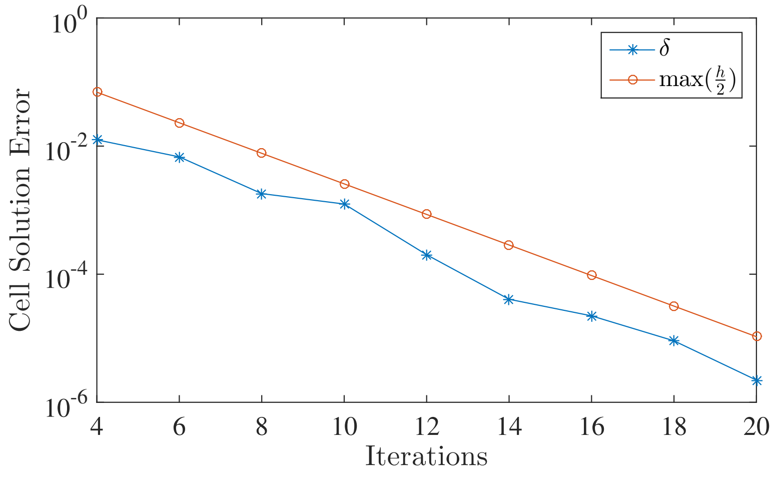

Fig. 2. The convergence of the solutions for Equation (8) as a function of iteration number of subdivisions. Circles: half of the largest cell dimensions and the upper bound of the error. Stars: the norms of the distance from the cell center to the corresponding true solution. Due to the $P-1$ property of the solution cells, the true solutions all locate within the cells found by the cell mapping method.

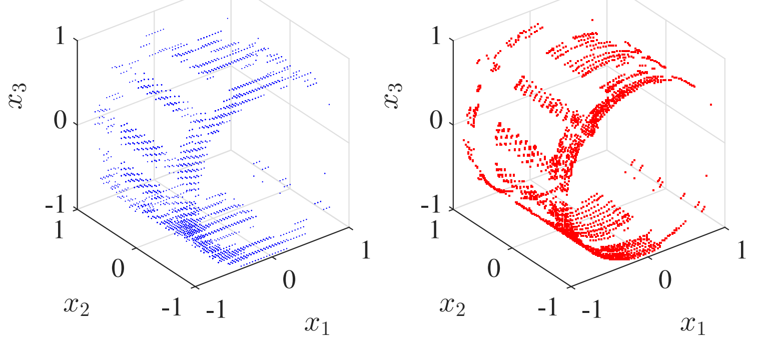

Fig. 3. The zeros of Equation (9). The initial partition of the space is 2^6. 18 iterations of subdivisions are taken. 11176 $P-1$ cells are found as the solutions. Left: Blue dots are the centers of the solution cells projected to a low dimensional space. Right: Red dots are the point-wise solutions computed by a search method starting from the centers of the solution cells. A fine global structure of the solution set is discovered. The error of the solutions obtained from the post point-wise search can be as small as 10^{-9}. The entire calculation takes 38.6 seconds.

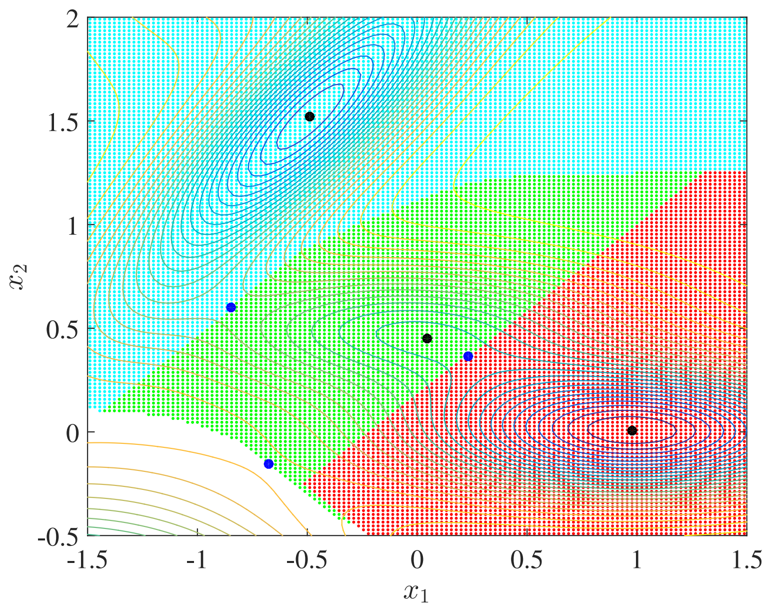

Fig. 4. The results of the global analysis of the two dimensional potential field in Equation (14). The cell space partition is 135×135. Different colors indicate the domains of attraction of the stable equilibria. Black dots are the stable equilibria. Blue dots are the saddles. The white area at lower left cornor is mapped out of the region of interest. The saddles are all located on the boundaries of domains of attraction.

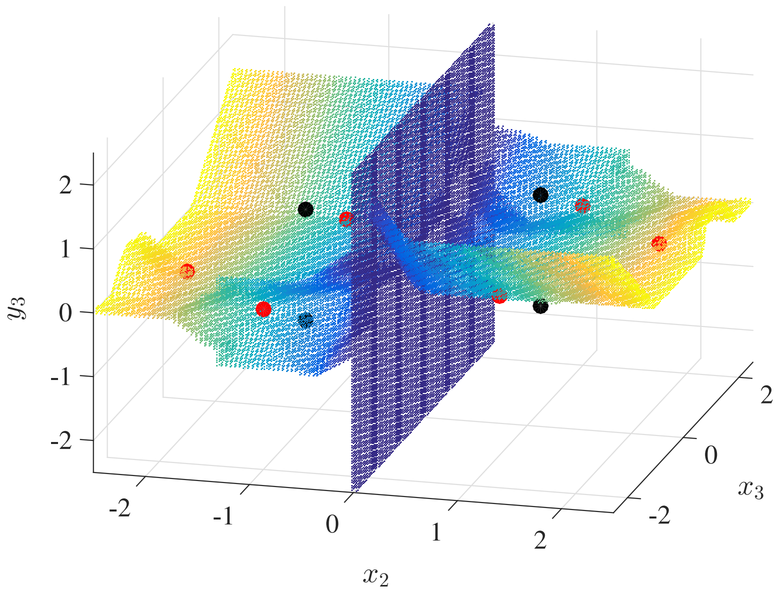

Fig. 5. The stability boundary of the $3D$ potential field. The boundary surfaces divide the $3D$ space into four parts. The surface is anti-symmetric with respect to plane $x_3 - y_3$. Plane $x_3 - y_3$ is also a part of the boundary. The color code of the irregular surface is scaled with respect to $|x_2|$. Black dots: the stable equilibrium points. Red dots: the saddles. The cell space resolution is $120 \times 120 \times 120$. The computational time is 3.8146 seconds. 82221 cells are found as the boundary cells.