The Adjoining Cell Mapping and Its Recursive Unraveling, Part I: Description of Adaptive and Recursive Algorithms

P. J. ZUFIRIA and R. S. GUTTALU

Abstract. A new type of cell mapping, referred to as an *adjoining cell mapping*, is developed in this paper for autonomous dynamical systems employing the cellular state space. It is based on an adaptive time integration employed to compute an associated cell mapping for the system. This technique overcomes the problem of determining an appropriate duration of integration time for the simple cell mapping method. Employing the adjoining mapping principle, the first type of algorithm developed here is an *adaptive mapping unraveling algorithm* to determine equilibria and limit cycles of the dynamical system in a way similar to that of the simple cell mapping. In addition, it is capable of providing useful information regarding the behavior of dynamical systems possessing pathological dynamics and of systems with rapidly changing vector field. The adjoining property inherent in the adjoining cell mapping method, in general, permits development of new recursive algorithms for unraveling dynamics. The required computer memory for a practical implementation of such algorithms is considerably less than that required by the simple cell mapping algorithm since they allow for a recursive partitioning of state space for trajectory analysis. The second type of algorithm developed in this paper is a *recursive unraveling algorithm* based on adaptive integration and recursive partitioning of state space into blocks of cells with a view toward its practical implementation. It can find equilibria of the system in the same manner as the simple cell mapping method but is more efficient in locating periodic solutions.

Key words: Nonlinear autonomous dynamical systems, simple cell mapping, limit cycles, adjoining cell mapping, adaptive integration, recursive algorithms.

1. Introduction

The method of cell-to-cell mapping is a general and powerful computational technique for locating the equilibrium points, limit cycles and periodic solutions of nonlinear dynamical systems, see [2, 3, 4]. The method discretizes the state space to create a mapping from a set of integers to a set of integers which contains essentially the underlying dynamics of a given system. Some questions arise in a practical implementation of the cell mapping technique which are concerned with the creation of an associated cell mapping of a dynamical system. To elaborate on this, consider the system of nonlinear autonomous differential equations given by

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}), \quad \mathbf{x} \in \mathbf{R}^N, \quad \mathbf{F}: \mathbf{R}^N \to \mathbf{R}^N, \quad \mathbf{F} \in C^0, \tag{1}$$

where $\mathbf{x}$ is an N-dimensional state vector and the vector field $\mathbf{F}$ is, in general, a nonlinear function of $\mathbf{x}$. In order to define an associated cell mapping, the system (1) is integrated for different initial conditions (corresponding to different cells in the cell state space) over a fixed time interval T_S. Since the duration of this time interval is arbitrarily chosen *a priori*, an important question is concerned with its selection. This question has not been addressed yet in the cell mapping literature, see [4]. The simple cell mapping method employs the same (constant) time duration T_S for every cell to obtain its image cell, see [3]. This presumes that the evolution of the local dynamics is very much the same throughout the state space. However, this is not necessarily true in general, for instance refer to the properties of certain autonomous systems studied in [1, 6, 7, 8]. The vector field $\mathbf{F}$ of such systems may change rapidly in the vicinity of singular manifolds and thus require a sophisticated numerical treatment for delineating their global behavior. In short, there may be systems for which no value of T_S provides meaningful results when the simple cell mapping method is employed. In addition, whenever an appropriate T_S exists, one can only find it intuitively or by trial and error.

The next question while implementing the simple cell mapping method is the exponential growth in the number of cells required as the dimension of the state space increases. This immediately demands the availability of a large amount of computer memory for its implementation since the entire cell mapping must be stored for a global analysis of the system. Some variations of the cell mapping technique have been studied to develop algorithms for locating stable invariant sets without having to store the entire cell mapping (see [4]). The main purpose of these algorithms is to refine the location of stable invariant sets possessed by a dynamical system but they have to be previously found by employing a large cell size.

This paper presents a refinement of the simple cell mapping technique referred to as an *adjoining cell mapping* technique. The central concept is the creation of an *adjoining cell mapping* where the integration time duration T is determined adaptively. It will be shown here that such a mapping is always guaranteed for dynamical systems of the type (1). The new method overcomes the problem of selecting an integration time when creating an associated cell mapping from a continuous-time dynamical system of the type (1). Two kinds of algorithms for practical implementation are considered here based on the adjoining mapping concept. First, based on an adaptive time integration, an *adaptive mapping unraveling algorithm* for the associated adjoining cell mapping will be presented to unfold the global behavior of the system. The unraveling of this new map can be performed in a manner quite similar to that for the simple cell mapping for locating periodic motions and the domains of attraction associated with them.

The use of adjoining mapping naturally permits development of new *recursive algorithms* for studying global dynamics of nonlinear systems. A fundamental feature of these algorithms is to partition the state space recursively into smaller domains such that a dynamic analysis can be performed in each of the domains independently of the other. The computer memory required for a practical implementation of such algorithms for locating equilibria and limit cycles of (1) is considerably less than that required by the simple cell mapping method. Although the amount of memory required is problem dependent, the amount of reduction is determined by the factor $n_{\max}/2$, where $n_{\max}$ is the maximum number of divisions performed on a component of the state variable. In conjunction with the adaptive mapping unraveling algorithm, a second kind of algorithm, referred to as a *recursive unraveling algorithm*, is developed in this paper with pragmatic considerations. An added advantage of this recursive algorithm is that it is capable of analyzing higher order dynamical systems. The recursive unraveling presented here for the adjoining cell mapping technique has certain similarities with the *state increment dynamic programming*, see [5].

The method of adjoining cell mapping determines equilibrium points of the system (1) in a manner very similar to that of the simple cell mapping method (i.e. with a similar performance). However, it is more effective in locating limit cycles (or periodic solutions). As opposed to the simple cell mapping method which replaces limit cycles by several groups of disconnected periodic cells, our method provides a single connected periodic group of cells for every limit cycle. In addition, the technique is capable of unraveling pathological dynamics, as in the case of dynamical systems of the form considered in [1, 6, 7, 8]. Application of the adjoining cell mapping formulation will be considered in an accompanying paper [9]. Its application to the generalized cell mapping formulation (see [4]) can provide a significant memory saving but this topic will not be considered here.

This paper is organized as follows. Section 2 presents some fundamental notions of the simple cell mapping method which are essential to introduce the concept of adjoining cell mappings. Section 3 deals with basic definitions regarding an adaptive integration procedure to generate an adjoining cell mapping. It also provides an adaptive mapping unraveling algorithm to unfold the map. Section 4 explains the recursive unraveling of an adjoining cell mapping. Comparative properties between the simple cell mapping and the adjoining cell mapping are considered in Section 5. Concluding remarks of this paper appear in Section 6.

2. Cell Mapping Dynamical Systems

In order to motivate the adjoining cell mapping method, we briefly discuss the *cell-to-cell mapping* computational approach for determining the global behavior of nonlinear autonomous dynamical systems. This involves determination of the location of equilibrium points and periodic solutions of (1). For asymptotically stable equilibrium points and periodic solutions, it is also required to find their corresponding domains of attraction. The computational procedure will be based on the idea of discretization of the state space to create a mapping. In this formulation, the state space is though of not as a continuum but instead as a collection of intervals referred to as *cells*. Each cell in the discretized state space or *cell state space* is to be treated as a state entity. The motivation for discretizing the state space in terms of cells is evidenced by the facts that in practice state variables can be measured only to a certain accuracy and that in digital computer computations of the state variables round off errors incur. The method of cell mapping applies to strongly nonlinear systems, and to both autonomous and nonautonomous systems. Since this method avoids repetitive time consuming calculation of the trajectories of dynamical systems, it has been found to be an efficient computational tool. The simple cell mapping algorithm is applicable to a compact state space. However, a compactification procedure may be employed to analyze the entire state space instead of a fixed state space, see [4] for further discussions.

In order to construct a cellularly structured state space S, let the coordinate axes of the state variables x_i, $i = 1, 2, \ldots, N$, be divided into a large number of intervals of uniform size h_i. The new state variable z_i along the x_i-axis is defined to contain all x_i such that

$$\left(z_i - \frac{1}{2}\right)h_i \le x_i < \left(z_i + \frac{1}{2}\right)h_i \,. \tag{2}$$

By definition z_i in (2) is an integer. A *cell vector* $\mathbf{z}$ is defined to be an N-tuple z_i, $i = 1, 2, \ldots, N$. Clearly, a point $\mathbf{x} \in \mathbf{R}^N$ with components x_i belongs to a cell $\mathbf{z} \in S$ with components z_i if and only if x_i and z_i satisfy (2) for all i. The space S consisting of elements which are N-tuple of integers is referred to as an N-dimensional *cell space*.

With this background, one can define a cell-to-cell mapping dynamical system in the form

$$\mathbf{z}(n+1) = \mathbf{C}(\mathbf{z}(n)) \,, \quad \mathbf{C}\colon S \to S \,, \quad n \in \mathbf{Z} \,, \tag{3}$$

where **C** is referred to as a *simple cell mapping* and is perceived of as a mapping from a set of integers to a set of integers. Equation (3) then describes evolution of a cell dynamical system in an N-dimensional cell state space S. For a detailed treatment concerning the properties of the map **C** and its other refinements, the reader is referred to the research monograph by Hsu [4]. In the following, we concentrate on simple cell mapping method only referring to it simply as cell mapping.

2.1. Creating an Associated Cell Mapping from an Autonomous Dynamical System

To obtain a cell mapping associated with the autonomous dynamical system (1), the following procedure may be employed. First, construct a cell space structure in the state space region with cell size h_i in the x_i-direction. Let $\mathbf{x}^d(t_0)$ denote the center point of the cell $\mathbf{z}(t_0)$ so that $x_i^d(t_0) = h_i z_i(t_0)$. Integrate the state equations (1) from time $t = t_0$ to time $t = t_0 + T_S$, where T_S is the total integration time. Suppose that the trajectory starting from $\mathbf{x}^d(t_0)$ terminates at $\mathbf{x}^d(t_0 + T_S)$. The cell in which $\mathbf{x}^d(t_0 + T_S)$ lies is taken to be $\mathbf{z}(t_0 + T_S)$, the image cell of $\mathbf{z}(t_0)$. Specifically,

$$z_i(t_0 + T_S) = C_i(\mathbf{z}(t_0)) = Int\left[\frac{x_i^d(t_0 + T_S)}{h_i} + \frac{1}{2}\right], \qquad i = 1, 2, \ldots, N, \tag{4}$$

where $Int\ [u]$, for any real number u, represents the largest integer such that $Int\ [u] \leq u$. This process of finding image cells is repeated for every cell in the cell state space S. The mapping **C** so obtained is a cell mapping associated with the dynamical system (1) using the *center point method* described above to compute the image cells. It should be noted that the associated cell mapping **C** of (1) is obtained by applying the same integration time T_S uniformly to all the cells in the cell state space S. The crucial step here is to choose a proper value of T_S to obtain meaningful results (for more discussion on this topic, see Section 3).

Once the mapping **C** in (3) is obtained, the crucial step in cell mapping analysis is to unravel the dynamic information of the original system (1) contained in (3) by examining the long time behavior of the *cell sequences*. A cell sequence is the trajectory of (3) starting from an initial cell state $\mathbf{z}(0)$ and is the set of integer cell sequence $\{\mathbf{z}(k)\}$, $k = 0, 1, 2, \ldots$. In practical applications, the state variables assume a finite range of values. Hence, one is usually interested in a fixed state space region which contains a finite number of cells even though the number of cells may be huge; the cells in this state space will be referred to as the *regular cells*. The complement of the fixed state space is referred to as the *sink cell*. Once a regular cell state of the system maps to the sink cell, its long time behavior is unknown and its motion is eventually locked in the sink cell. An important property of the cell sequences, due to the finite number of cells in the state space, is that all sequences must terminate with a finite number of cell mappings into one of the steady cell states: equilibrium cells, periodic cells, or the sink cell. This is the key to the simple cell mapping algorithm for global analysis described by Hsu and Guttalu [3] from which the following characteristics of the dynamics of the system (1) can be obtained all at once:

(1) Location of the equilibrium states and periodic solutions in a given state space region.
(2) Domains of attraction associated with the asymptotically stable equilibrium states and periodic solutions.
(3) Step-by-step evolution of the global behavior of the system starting from any initial state within the cell state space.

The associated cell mapping **C** in equation (4) is viewed as an approximation of the original dynamical system (1). The degree of approximation can be improved by simply reducing the cell size h_i (or equivalently, increasing the number of cells in S). The cell mapping **C** as defined by (3), in general, replaces the stable equilibrium points and stable periodic solutions of (1) in the state space $\mathbf{R}^N$ with sets of periodic cells in the cell state space S; unstable equilibrium points and unstable periodic solutions of (1) may not be recovered. Each set of periodic cells may consist of a group of "true periodic cells" representing the original periodic motion of the dynamical system (1) and groups of "pseudo-periodic cells" which surround the true periodic cells. These pseudo-periodic cells are the result of discretization of the state space; the state space occupied by them has been shown to decrease as the cell sizes are made smaller, see [3].

3. Adjoining Cell Mapping and Adaptive Time of Integration

When applying the simple cell mapping method to the autonomous system (1), three important questions arise. The first question is concerned with the problem of choosing an appropriate integration time T to obtain the mapping **C** in (4). Too small a time step usually results in a large number of pseudo-periodic cells (which, in this case, are cells mapping to themselves) making it difficult to identify the true periodic cells. On the other hand, a longer integration time step would increase the computational time in creating the mapping **C** and may result in a fewer number of periodic cells associated with a periodic motion (for example, only a few cells may replace a stable limit cycle). Hence, in general, the simple cell mapping method provides useful information regarding global dynamics but requires a very careful and intuitive selection of an integration duration T_S. There is an implicit trade-off between the cell size h_i and T_S. For a given partition of state space (fixed h_i), one can select an optimal value of T_S based on the average time that trajectories need to evolve a distance of the order of the size of a cell. This optimal integration time T_S as a function of cell size h_i is very difficult to determine *a priori*. In addition, when analyzing systems with rapidly changing vector field **F**, it may happen that simple cell mapping may not provide meaningful results no matter what value of T_S is employed.

The second question is the exponential increase in the number of cells in the cell state space S as the dimension of the system increases. This directly limits the practical utility of the cell mapping method to higher order systems since computer memory required for the implementation of the global cell mapping algorithm also increases exponentially. This problem is inherent to all higher dimensional systems. Finally, the third question is related to unstable solutions which may not be found by using the simple cell mapping method alone. However, other techniques based on the simplicial mapping, cell functions, and the construction of appropriate dynamical systems have been developed to address this issue; for details, the reader is referred to [1, 4, 6, 7, 8].

The first two questions of selecting an appropriate integration time T and of handling large number of cells are addressed here. With the aid of an adaptive integration scheme, a mapping of the form

$$\mathbf{z}(n+1) = \mathbf{C}_A(\mathbf{z}(n)), \quad \mathbf{C}_A: S \to S, \quad n \in \mathbf{Z} \tag{5}$$

for the autonomous dynamical system (1) can be found. The mapping $\mathbf{C}_A$ is referred to as an *adjoining cell mapping*, or simply as an *adjoining mapping*. By creating an adjoining cell mapping, it will be shown later that the two questions mentioned can be simultaneously considered.

3.1. *Adjoining Cell Mapping*

In order to develop the concept of an adjoining cell mapping, we need the following two definitions which are taken from Hsu [4].

DEFINITION 1. The distance between any two cell vectors $\mathbf{z}$ and $\mathbf{z}'$, denoted by $d(\mathbf{z}, \mathbf{z}')$, is defined as

$$d(\mathbf{z}, \mathbf{z}') = \max_{1 \le i \le N} |z_i' - z_i| . \tag{6}$$

DEFINITION 2. Two cell vectors $\mathbf{z}$ and $\mathbf{z}'$ are said to be adjoining if and only if

$$d(\mathbf{z}, \mathbf{z}') = 1 . \tag{7}$$

When an associated simple cell mapping of (1) is obtained by direct integration, the same constant integration time T is applied to every cell in the cell state space S. Every cell can, *a priori*, map to any cell in S. That is, for a fixed T, the distance $d(\mathbf{z}(t_0), \mathbf{z}(t_0 + T))$ can be very large depending on how "fast" trajectories evolve in the corresponding region of the state space, see Figure 1 where $t_0 = 0$ and $T = T_S$. The adjoining cell mapping defines a cell mapping for which the distance between any cell and its map is less than or equal to one. Since trajectories of the system (1) are continuous in state space, they may be best approximated by a cell mapping which preserves such a "continuity" property. In the cell state space, this property corresponds with an adjoining mapping. Keeping this in mind, we proceed to examine the evolution of trajectories of the autonomous system (1) in the partitioned state space (i.e., the cellular state space). If the vector field $\mathbf{F}$ is continuous, these trajectories are continuous in the state space and pass from one cell to another. It is important to characterize the instants of time at which such a transition occurs.

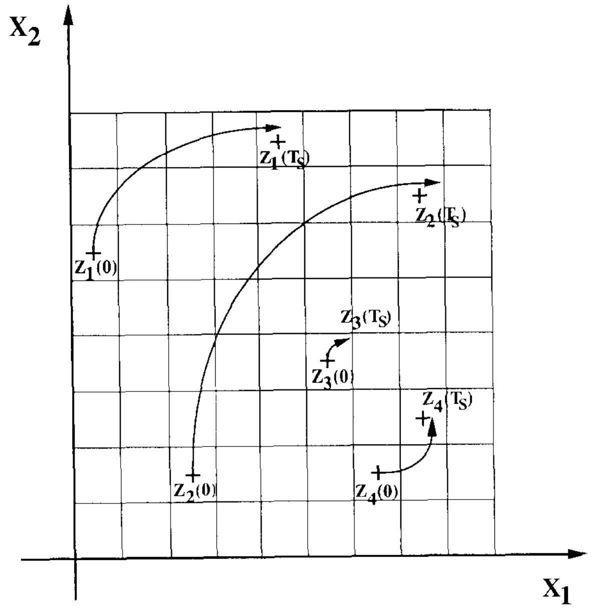

Fig. 1. Determination of an associated simple cell mapping with a constant duration of integration $T = T_S$.

DEFINITION 3. A *transition time* t_t is a time instant such that for any small enough neighborhood B_t of t_t it satisfies the condition

$$\max_{t_1, t_2 \in B_t} d(\mathbf{z}(t_1), \mathbf{z}(t_2)) = 1 . \tag{8}$$

Note that, accordingly, a transition time t_t satisfies the relation $x_i(t_t) = (z_i(t_t) - \frac{1}{2})h_i$ for some $1 \le i \le N$. In others words, a transition time is either the last instant of time at which a trajectory is within a given cell or the first instant of time at which the trajectory is in a new cell. We now state the following remark.

REMARK 1. *Every trajectory of the dynamical system (1) starting inside a cell* $\mathbf{z} \in S$ *either remains inside it forever or transits to an adjoining cell at a transition time.*

A transition time t_t for a cell can be determined by separately analyzing each state space variable as follows. Let $\mathbf{x}(t_0)$ be an initial state in $\mathbf{R}^N$ located in the cell $\mathbf{z}(t_0)$ with components $(z_1(t_0), z_2(t_0), \dots, z_N(t_0))$. Let $\mathbf{z}(t)$ be the cell containing the trajectory $\mathbf{x}$ at a given time t. For $t > t_0$, consider the time evolution of a component $z_i(t)$ of the cell $\mathbf{z}(t)$ for $i = 1, 2, \dots, N$. From equation (4), it is clear that

$$|z_i(t+\epsilon) - z_i(t)| \le 1 , \quad \forall t \ge t_0 \tag{9}$$

for small enough $\epsilon > 0$. Consequently, either $z_i(t) = z_i(t_0)\ \forall t_0$ or there exists a transition time $t_i > t_0$ such that $|z_i(t) - z_i(t_0)| = 1$ for some $t \in B_{t_i}$ (where B_{t_i} is any small neighborhood of t_i) and $|z_i(t) - z_i(t_0)| = 0\ \forall t_0 \le t < t_i$. If the former case happens, we will consider the time t_i to be infinity.

Considering all the components of $\mathbf{z}(t_0)$ together, we define $t_t = \min_i\{t_i\}$. If $t_i = \infty\ \forall i$ then t_t is also infinite, meaning that the trajectory starting in the cell $\mathbf{z}(t_0)$ remains in $\mathbf{z}(t_0)$ forever. In this case, the trajectory remains in the initial cell state forever. If t_t is finite, then

$$d(\mathbf{z}(t_t), \mathbf{z}(t_0)) = \max_{1 \le i \le N} |z_i(t_t) - z_i(t_0)| = 1 \quad \text{for some } t \in B_{t_t} ,$$

$$d(\mathbf{z}(t), \mathbf{z}(t_0)) = \max_{1 \le i \le N} |z_i(t) - z_i(t_0)| = 0 , \quad \forall t_0 \le t < t_t .$$

In this case there exists a transition time $t_t > t_0$ at which the trajectory starting from $\mathbf{z}(t_0)$ transits to a cell $\mathbf{z}(t_t)$ which is adjoining to the initial one.

The above remark is intuitively obvious since trajectories of the autonomous system (1) are continuous in the state space. However, it clearly establishes two possible behaviors of a trajectory starting in a cell. This property may be exploited in defining a new type of cell mapping to enhance computational aspects of nonlinear dynamical systems. Motivated by the above remark, we define the cell adjoining time and integration time as follows.

DEFINITION 4. An *adjoining cell-time* or simply a *cell-time* associated with a given cell $\mathbf{z} \in S$ is any time t_z such that

$$d(\mathbf{z}(t_z), \mathbf{z}(t_0)) = 1 . \tag{10}$$

DEFINITION 5. A *cell adjoining integration time* or simply a *cell integration time* associated with a given cell $\mathbf{z} \in S$ is any time t_c such that

$$d(\mathbf{z}(t_0 + t_c), \mathbf{z}(t_0)) = 1 , \quad t_c = t_z - t_0 . \tag{11}$$

We are interested in finding an adjoining cell-time for every cell in the cell state space S to define a different kind of cell mapping. This mapping will be referred to as an *adjoining cell mapping* expressed by (5) which now characterizes the dynamics of the original system (1). It should be noted that a cell-time t_z (and consequently, a cell integration time t_c) associated with a particular cell depend on the cell itself and hence they are, in general, different for each cell in the cell state space. Besides, such times are not unique for a given cell. It should be noted that if a trajectory of (1) starting in a cell transits into an adjoining cell, then there will be a time interval (may be just a point, the transition time) during which this trajectory remains within the adjoining cell.

DEFINITION 6. An *adjoining time interval* for a given cell $\mathbf{z}$ is an interval I_A given by $I_A = [t_z, t_z + \delta]$, $\delta \geq 0$ such that $|\mathbf{z}(t) - \mathbf{z}(t_0)| = 1$, $\forall t \in I_A$.

In the next section we outline a procedure to determine a time $t \in I_A$ for a cell in cell state space. The algorithm provided there makes use of the following remark which shows that when a mapping to a nonadjoining cell (a cell that is neither adjoining nor itself) is found, a whole interval I_A (not just an instant of time) is guaranteed to exist.

REMARK 2. *Suppose that at time* t_1, $d(\mathbf{z}(t_1), \mathbf{z}(t_0)) = 0$ (*that is, the trajectory is in the starting cell* $\mathbf{z}(t_0)$) *and that at time* $t_2 > t_1$, $d(\mathbf{z}(t_2), \mathbf{z}(t_0)) > 1$ (*trajectory is in a nonadjoining cell*). *Then, there is at least one time interval* $I_A = (t_a, t_b)$ *such that* $I_A \subset (t_1, t_2)$, $t_a > t_b$ *and* $d(\mathbf{z}(t), \mathbf{z}(t_0)) = 1\ \forall t \in I_A$ (*that is, the trajectory is in an adjoining cell*).

It is clear from Remark 1 that the trajectory in the cell $\mathbf{z}$ at time t_1 must pass through an adjoining cell to map to a nonadjoining one at time t_2. This means that some component x_i must change by at least a value $\Delta x_i = h_i > 0$ within the adjoining cell. Since $\mathbf{F} \in C^0$ in the cell, $\mathbf{F}(\mathbf{x})$ is bounded at any interior point of the cell and there exists a time $\Delta t > 0$ which is the time needed for the trajectory to undergo Δx_i variation. If t_a is the transition time when the trajectory enters the adjoining cell, then $t_b = t_a + \Delta t > t_a$ is the transition time when the trajectory leaves it. Hence, as $t_a > t_1$ and $t_b < t_2$, this interval must be included in (t_1, t_2).

The significance of this remark is illustrated in Figure 2 which represents a cellular state space for a two-dimensional state space. The cell $\mathbf{z}$ under consideration is in the center of the region and its adjoining cells are shaded. To illustrate the ideas developed in this section, a trajectory starting from the center point of the cell $\mathbf{z}$ at time $t = t_0$ is indicated. Note that both $\mathbf{x}(t_0)$ and $\mathbf{x}(t_1)$ are in the initial cell $\mathbf{z}$ whereas $\mathbf{x}(t_2)$ is in a nonadjoining cell. Then, Remark 2 guarantees the existence of a time $\hat{t}$ such that $\mathbf{x}(\hat{t})$ will be in a cell adjoining to $\mathbf{z}$. Since in this case the trajectory passes by two adjoining cells, say at times $\hat{t}_a$ and $\hat{t}_b$, the nonuniqueness of adjoining cell-time is clearly illustrated.

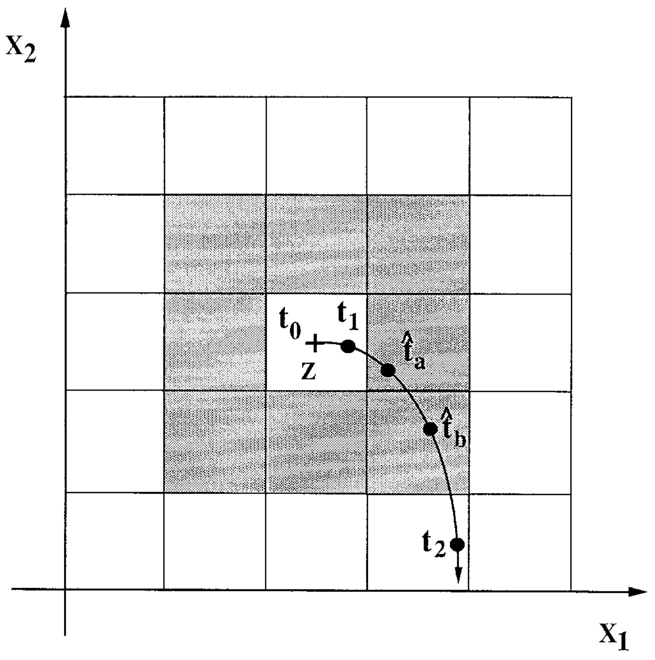

Fig. 2. Determination of cell adjoining time and an associated adjoining cell mapping.

3.2. Adjoining Cell-Time Finding Algorithm

The crucial ingredient required for defining an adjoining cell mapping is the determination of an adjoining cell-time for each cell in the cell state space. The following algorithm provides an adjoining cell-time and the corresponding adjoining mapping for a given cell.

We begin by taking any cell $\mathbf{z}_0$ which belongs to the state space under consideration. The aim of the algorithm is to find an adjoining cell-time t_z such that the trajectory starting at the center point of the cell $\mathbf{z}(t_0) = \mathbf{z}_0$ at time t_0 is in the cell $\mathbf{z}(t_z)$ at time t_z which is adjoining to $\mathbf{z}_0$. Finding t_z means that an an adjoining mapping $(\mathbf{z}(t_z) = \mathbf{C}_A(\mathbf{z}(t_0))$ has been determined.

The adjoining cell-time t_z can be obtained following the two steps described below:

STEP I: The aim of Step I is to find an interval (t_1, t_2) as defined in Remark 2, such that $\mathbf{z}(t_1) = \mathbf{z}_0$ and $\mathbf{z}(t_2)$ are nonadjoining cells. During the search procedure an adjoining cell-time t_z might be found, in which case the aim of the entire algorithm would have been accomplished and Step II is not needed. For the reasoning followed in Step I such a possibility will be obviated.

Since we can select $t_1 = t_0$, only t_2 remains to be found. Choosing the center point of $\mathbf{z}(t_1)$ as initial condition, a numerical integration of the system (1) is performed from t_1 to different prescribed times $t_c > t_0$ and the location of $\mathbf{z}(t_c)$ is checked until $\mathbf{z}(t_c = t_2)$ is nonadjoining to $\mathbf{z}(t_1)$.

To determine such an appropriate time t_c, we begin with arbitrary checking times $t_c = t_0 + kT$ (for increasing values of $k = 1, 2, \ldots$) where T has been previously specified. For each value of k, two possibilities exist:

(a) $\mathbf{z}(t_0 + kT) = \mathbf{z}(t_1)$ (trajectory at time $t_0 + kT$ is still in the starting cell). In this case we increment the value of k by 1 and check the trajectory at the new time $t_c = t_0 + kT$. This procedure is repeated by incrementing k until $\mathbf{z}(t_c = t_0 + kT)$ is nonadjoining cell to $\mathbf{z}(t_c =$

$t_0 + (k-1)T) = \mathbf{z}(t_1) = \mathbf{z}_0$ (in which case we are now in possibility (b) explained below), or when a prescribed limit of integration time is surpassed ($t_e = t_0 + kT > t_{\max}$) and still $\mathbf{z}(t_e) = \mathbf{z}(t_1)$.

If the time $t_e = t_0 + kT$ surpasses $t_{\max}$, then the interval (t_1, t_2) as defined in Remark 2 is not found. Hence, no cell-time can been determined and we consider that cell $\mathbf{z}_0$ maps to itself indicating that it is an equilibrium cell. The algorithm stops here.

(b) $d(\mathbf{z}(t_0 + kT), \mathbf{z}(t_1)) > 1$ (trajectory at $t_e = t_0 + kT$ is in a nonadjoining cell). The interval (t_1, t_2) with $t_1 = t_0 + (k-1)T$ and $t_2 = t_0 + kT$ has been found.

STEP II: The goal of Step II (and of the overall algorithm) is to find an adjoining cell-time t_z. This step relies on the fact that the interval (t_1, t_2) with $t_1 = t_0 + (k-1)T$ and $t_2 = t_0 + kT$ has been found (satisfying the required conditions of Remark 2). We then evaluate the mapping at the middle of the time interval given by $\hat{t} = (t_1 + t_2)/2$. If $\mathbf{z}(\hat{t})$ is adjoining to $\mathbf{z}_0$, the aim of the algorithm is accomplished with adjoining cell-time $t_z = \hat{t}$. Otherwise, we partition the interval (t_1, t_2) into two corresponding half-time intervals $(t_1, \hat{t})$ and $(\hat{t}, t_2)$, and perform the following selection:

(a) If $\mathbf{z}(t_1) = \mathbf{z}(\hat{t})$, the time interval $(\hat{t}, t_2)$ is now considered. This new time interval satisfies also the required conditions of Remark 2. Hence, Step II of this algorithm is recursively applicable.

(b) If $\mathbf{z}(t_1) \neq \mathbf{z}(\hat{t})$, the interval $(t_1, \hat{t})$ is considered. Here again, this new time interval has the required properties for a recursive application of Step II of the algorithm.

The algorithm will successfully end when a middle point time is checked for which the trajectory is in an adjoining cell.

It will be shown later that the above algorithm always concludes with a finite number of steps. We now state the algorithm in a more precise mathematical formulation.

Adaptive algorithm for obtaining an associated adjoining cell mapping:

1. Fix the integration times by choosing $t_1 = 0$, $t_2 = T < t_{\max}$ (where $t_{\max}$ is a specified maximum integration time) and a *time interval size* $\Delta t_k = T$, for $k = 1$.
2. Let $\mathbf{z}$ be a cell whose adjoining mapping is to be determined. Take the initial condition to be the center point of the cell $\mathbf{z}$ and integrate (1) from time $t = 0$ to $t = t_2$ to find its image cell $\mathbf{z}'$.
   (i) If $\mathbf{z}'$ is an adjoining cell, stop. The adjoining mapping of the cell has been found and is $\mathbf{z}' = \mathbf{C}_A(\mathbf{z})$.
   (ii) If $\mathbf{z}'$ is neither an adjoining cell nor equal to $\mathbf{z}$, then define $\Delta t_{k+1} = \Delta t_k/2$ and $t_2 = t_2 - \Delta t_{k+1}$. Restart step 2 with the new integration time t_2.
   (iii) If $\mathbf{z} = \mathbf{z}'$, then there are two possibilities and we define

$$\Delta t_{k+1} = \begin{cases} T, \text{if } \Delta t_k = T \, , \\ \dfrac{\Delta t_k}{2} \, , \text{if } \Delta t_k < T \, . \end{cases} \tag{12}$$

   Then we take $t_1 = t_2$ and $t_2 = t_2 + \Delta t_{k+1}$. If $t_2 < t_{\max}$, go back to the beginning of step 2 with integration time t_2. Otherwise, if $t_2 > t_{\max}$ the cell maps to itself.
3. Repeat steps 1 and 2 for each cell in S.

If the algorithm does not find an interval (t_1, t_2) which satisfies the conditions stated in Remark 2, the cell is said to map to itself. Finally, we provide Remark 3 below to point out that if the interval (t_1, t_2) satisfying the conditions of Remark 2 is found, then the algorithm provides an adjoining cell-time and an associated adjoining mapping.

REMARK 3. *If the above algorithm finds a time $t_2 < t_{\max}$ at which the trajectory is in a cell nonadjoining to $\mathbf{z}(t_1)$ then the algorithm, in a finite number of iterations, finds a time t^* such that $t_1 < t^* < t_2$, and $\mathbf{z}(t^*)$ is adjoining to $\mathbf{z}(t_1)$.*

From Remark 2, it follows that between any pair of points $[t_1, t_2]$ we can define at least one (maybe many, but a finite number) intervals I_i^*, $i = 1, 2, \ldots, n$, satisfying the following:

(i) $\forall t^* \in I_i^*$, $\mathbf{z}(t^*)$ is adjoining to $\mathbf{z}(t_1)$ (i.e., $d(\mathbf{z}(t^*), \mathbf{z}(t_1)) = 1$).
(ii) If we define $t_i^m = \min I_i^*$, $t_i^M = \max I_i^*$, and any corresponding neighborhoods N_i^m and N_i^M, then there exist times $t_a \in N_i^m$ and $t_b \in N_i^M$ satisfying $\mathbf{z}(t_a) = \mathbf{z}(t_1)$ and $d(\mathbf{z}(t_b), \mathbf{z}(t_1)) > 1$.
(iii) The intervals I_i^* are the largest connected time intervals satisfying (i) and (ii) above.

Hence, the intervals I_i^* are time intervals at which the trajectory is in an adjoining cell after evolving from the initial cell and is on its way towards a nonadjoining cell. We can now define the nonempty set $S^* = \cup_i I_i^* \subset [t_1, t_2]$. Since the vector field $\mathbf{F}$ is bounded, the size of I_i^* has a lower bound which guarantees that S^* is composed of a finite number of them. Let $\delta(I_i^*)$ be the size of I_i^* and $0 < \delta^* = \min_i \delta(I_i^*)$.

Step II of the algorithm is based on the recursive partition of a time interval $I = [t_1, t_2]$ by its middle point $\hat{t} = (t_1 + t_2)/2$ and the evaluation of the trajectory at time $\hat{t}$. This procedure generates a sequence of nested intervals $I_l = [t_1, t_2]_l$ with size $\delta(I_l) = d_l = d((t_1)_l, (t_2)_l)$ and a sequence of middle points $\hat{t}_l$ for $l = 1, 2, \ldots$. Note that $d_{l+1} = d_l/2$ and that $I_{l+1} \subset I_l$. From Remark 2, it follows that the set $S_l^* = S^* \cap I_l$ is nonempty $\forall l$.

Let us suppose that the algorithm never finds a time t^* (i.e., $\hat{t}_l \notin S_l^*$, $\forall l$). Hence, both $(t_1)_l$ and $(t_2)_l \notin S_l^*$, $\forall l$, and there should always be some $I_i^* \in S_l^*$ whose size would be smaller than d_l. But $\exists l_0$ such that $\forall l \geq l_0$, $d_l < \delta^*$, which contradicts our supposition.

Therefore, we conclude that $\exists l \leq l_0$ such that either $(t_1)_l$ or $(t_2)_l \in S_l^*$ and hence, $\exists l \leq l_0$ such that $\hat{t}_l \in S_l^*$ (i.e., we find a time t^* in a finite number of iterations).

3.3. Selection of Initial Integration Time Duration T

The adjoining cell mapping has been motivated to overcome the problem of determining an appropriate integration time (referred to as t^* in Remark 3) as well to handle systems with rapidly changing vector field $\mathbf{F}$. So far, the determination of t^* for each cell has been reduced to choosing an initial guess referred to as T in the algorithm. The influence of T is minor but sometimes noticeable in obtaining a map $\mathbf{C}_A$. One can increase the adaptability of the algorithm to find $\mathbf{C}_A$ by selecting T_i for a cell $\mathbf{z}_i$ also in an adaptive manner. That is, T_i for a given cell can be chosen to be t^* found for the previous cell $\mathbf{z}_{i-1}$. This way, only an initial guess T needs to be provided for just one cell $\mathbf{z}_0$ in state space. The effect of determining T in this manner is then negligible in obtaining a map $\mathbf{C}_A$ (which is now the outcome of a truly adaptive algorithm). In addition, a computational implementation of the algorithm will be significantly faster since for many cells T_i will provide an adjoining mapping (i.e., it is a good initial guess) avoiding an iterative search. This is because a mapping $\mathbf{C}_A$ is computed for cells which are arranged in an organized way (in the same way as for a map **C** in [3]) and one can expect that consecutive cells (cells adjoining to each other) have similar cell adjoining integration time.

3.4. Alternative Simplified Approaches

When formulating the algorithm for determining an adjoining mapping in Section 3.2, different time parameters T and $t_{\max}$ have been defined for the sake of generality. If we consider the case for which $T = t_{\max}$, the algorithm reduces to the following two steps:

(i) A simple cell mapping is defined for every cell in the region of interest with integration time T.
(ii) A new mapping is defined for cells which map to a nonadjoining cell.

Note that for the remaining cells, the simple cell mapping is already an adjoining cell mapping and it is preserved in that manner. So far step (ii), the major step for defining the adjoining cell mapping, has been based on finding an adjoining time interval. Here, we propose alternative algorithms to perform the steps in a more simplified (and, consequently, in a less accurate) manner.

A first assumption to simplify step (ii) is to consider that a trajectory in state space has evolved from $\mathbf{x}(t_0)$ to $\mathbf{x}(T)$ along a straight line (i.e., the vector field preserves proportionality among its components). This is to be implemented in step (i) appropriately. One can define an adjoining mapping by performing a geometric interpolation between $\mathbf{x}(t_0)$ and $\mathbf{x}(T)$ (supposedly $\mathbf{x}(t^*)$) so that the trajectory falls within a cell adjoining to the initial cell $\mathbf{z}(t_0)$. Based on this geometric reasoning, an adaptive search for the cell adjoining time is not needed here (thus avoiding a repetitive integration of the system) and the resulting algorithm is faster than the one presented previously.

A second and a more simplifying implementation concerns both steps (i) and (ii) and results by noting that a trajectory of the dynamical system (1) can be obtained using a variety of numerical integration schemes. If Euler one-step numerical integration scheme (the simplest one) is used, the function **F** needs to be evaluated only once. Based again on a geometric reasoning, the orientation and magnitude of the vector field **F** can determine whether the trajectory stays in the same cell or maps to an adjoining cell. Since an adjoining mapping can be computed directly in this case, neither a numerical integration of the system nor an adaptive search for a cell-time is needed. In general, the results obtained with this very simplified scheme are quite similar to the ones obtained through other procedures. In addition, the simplification leads to a very efficient scheme for obtaining an adjoining mapping (which is even several times faster than the simple cell mapping method).

3.5. Remarks on the Adjoining Mapping

It is obvious from previous sections that the adjoining mapping $\mathbf{C}_A$ and the corresponding algorithm for its determination are not unique. Besides the above mentioned algorithms, one can establish several other algorithms to select an adjoining mapping based on different criteria and intuition to better mimic the dynamics of the system (1). For example, for a given cell one can select its adjoining cell to be the last adjoining cell before its trajectory transits to a nonadjoining one. However, we will not pursue this topic further here. The main purpose of this paper is to point out the possibility of defining an adjoining mapping and to provide appropriate algorithms to demonstrate its utility for analyzing nonlinear autonomous systems.

Once an associated adjoining mapping $\mathbf{C}_A$ is obtained for a given dynamical system using the adaptive mapping algorithm presented earlier, then it only remains to reveal the dynamics contained in $\mathbf{C}_A$. Since (5) is of the same form as the simple cell mapping (3), the same global unraveling algorithm provided in [3] can be utilized for locating equilibrium cells, periodic cells and for obtaining their appropriate domains of attraction. For more details, see [4]. The resulting algorithm for unraveling the mapping $\mathbf{C}_A$ will be referred to as an *adaptive mapping unraveling algorithm*. In the next section, we present a different approach for unraveling the adjoining cell mapping (5) which will be based on the recursive partition of state space and the adjoining property of the map.

4. Recursive Unraveling of Adjoining Cell Mapping

This section presents an unraveling algorithm which is applicable to an adjoining mapping given by (5). The unraveling of the map (5), henceforth referred to as a *recursive unraveling algorithm*, is quite different in approach to the one described in [3] for the mapping $\mathbf{C}$ in (3) and it relies on the fact that a mapping in (5) is such that $d(\mathbf{z}, \mathbf{C}_A(\mathbf{z})) \leq 1$ for all $\mathbf{z} \in S$. The algorithm is motivated by the fact that the state space under consideration can be partitioned into subregions since the adjoining cell mapping of a cell is always to its neighbor. The partitioning is performed in such a manner that the dynamics in each partition via (5) can be analyzed independently of the dynamics in other partitions. For each partition, the new algorithm inherently makes use of the adaptive algorithm as described earlier for obtaining an adjoining map. It differs from that algorithm in the unraveling process in that it employs a recursion procedure for partitioning state space and for unraveling the resulting adjoining cell mapping for the entire state space under consideration. Since smaller partitions can be analyzed, in general, this new algorithm requires much less memory storage for its implementation than the simple cell mapping method. It provides the equilibria and limit cycles of the dynamical system being studied with a computational performance similar to that of the simple cell mapping method.

We begin by introducing the following definitions which will be employed for describing the recursive unraveling algorithm.

DEFINITION 7. A block is a set of cells in the cell state S with the following property: Suppose that two cells belong to the same block. Then, there is always a line in the state space connecting these two cells such that all the cells lying on this line also belong to the same block (that is, a block satisfies the property of connectedness).

It is reasonable (but not mandatory) to impose an additional condition such that a block does not have "holes" in its interior: For any two cells which do not belong to a block, there is always a line in the state space connecting these two cells such that all the cells lying on this line neither belong to the block. The two conditions above taken together would then guarantee that a block is simply connected.

The following two definitions, taken from set theory, also apply to a block when it is considered as a set of cells.

DEFINITION 8. Two blocks are *disjoint* if they do not have a common cell.

DEFINITION 9. The *union* of two or more blocks is a set (may be a block) containing all of their cells.

Usually the algorithm presented in this section groups blocks in such a way that they form a new larger block. Once a block is defined, the cells belonging to it can be classified based on the following definitions.

DEFINITION 10. A cell belonging to a given block is a *boundary cell* if it has at least one adjoining cell that does not belong to the block.

DEFINITION 11. A cell belonging to a given block is an *interior cell* if all of its adjoining cells belong to the block.

DEFINITION 12. A cell belonging to a given block is a *transition cell* if it is an interior cell, or if it has only one adjoining cell which does not belong to the block and this adjoining cell is the sink cell.

DEFINITION 13. A cell belonging to a given block is a *jump cell* if it is not a transition cell.

The following definitions have a dynamic character and are related to the information required by the unraveling algorithm at different stages.

DEFINITION 14. A cell is *active* if the information regarding where it maps is still required by the algorithm.

DEFINITION 15. A cell is *inactive* if the information regarding where it maps is not required any more by the algorithm.

The main purpose of the last two classification of cells is that information regarding inactive cells need not be stored in memory. Information regarding active cells only needs to be kept in memory as long as it is required by the algorithm (i.e., until they become inactive cells).

4.1. Basic Steps for Recursive Unraveling Algorithm

Suppose that a cellularly structured state space has been defined according to equation (2). Motivated by the above definitions, the basic steps of the unraveling algorithm are as follows:

1. Partition the cell state space into groups of disjoint blocks whose union is the entire state space region under consideration.
2. Each block is analyzed independently of the other. The dynamic analysis provides the following information.
   (a) Inside each block, equilibrium points and periodic solutions are located (for instance by employing the simple cell mapping algorithm presented in [4]). All the periodic cells and other cells in the block which map to them become inactive cells.

(b) Every sequence of cell mappings that cross through a block (starting from one jump cell, going may be through transition cells, to another jump cell and leaving the block) are characterized by just a mapping between the jump cells of the sequence. The intermediate transition cells become inactive cells.

(c) The boundary cells which map to a cell outside the block are kept for further consideration. Their mapping will be found in posterior analysis.

From the above it follows that it is only necessary to keep the information which is required for the location of any possible periodic solution that belongs to more than one block.

3. The analysis of trajectories that evolve through more than one block is carried out via a new study of the mapping corresponding to jump cells which remain from the previous analysis of each block.

4.2. Recursive Approach

Here, we elaborate on the recursive structure of the adjoining simple cell mapping algorithm. If desired, a new partition can be implemented in the analysis of a block in a recursive manner. Once a block is partitioned into a set of smaller ones, the procedure to follow for analyzing each small block is essentially the same as that for the large one. Based on this recursive approach, the following concepts are particularly important for the unraveling algorithm.

DEFINITION 16. The *level of partition* is the number of partitions that have been defined since the analysis of the total state space region under study was started.

The levels of partition define an ordering among all the partitions. Usually, the partitions are ordered from larger block partitions to smaller block partitions. A new partition level is defined when we decide to partition each block under study into a new set of smaller blocks. Related to the levels of partition we can also define the levels of analysis as follows.

DEFINITION 17. The *level of analysis* is the number of analysis since the first analysis in the lower (last) level of partition.

The levels of analysis define an ordering among all the analysis done at each partition level. The recursive structure makes these levels to be ordered from analysis on the smaller blocks to analysis on the bigger ones. Each level of analysis is associated with a particular level of partition. However, the levels of analysis are developed in reverse ordering than the levels of partition. A new higher analysis level is reached when we group a set of already analyzed blocks into a new bigger block for its global analysis, So, a level of analysis can also be called a *grouping level.* The structure of recursive block partitioning described by the levels of partition and analysis is very suitable for parallel processing implementation which will not be addressed here.

Figure 3 illustrates different partition and analysis levels for a given region (or block) under analysis. The region is partitioned at a first level into four blocks (B_1 to B_4) which are shown here separated by thick lines intersecting at the origin. Note that jumps cells corresponding to block B_4 are shaded. Each of these four blocks is partitioned in turn into another set of four blocks which is now associated with the second partition level. This second partition level corresponding to block B_1 is also illustrated by shading the blocks accordingly (B_{13} is one such block containing 2×2 cells). The blocks obtained at the second partition level can be analyzed independently (second

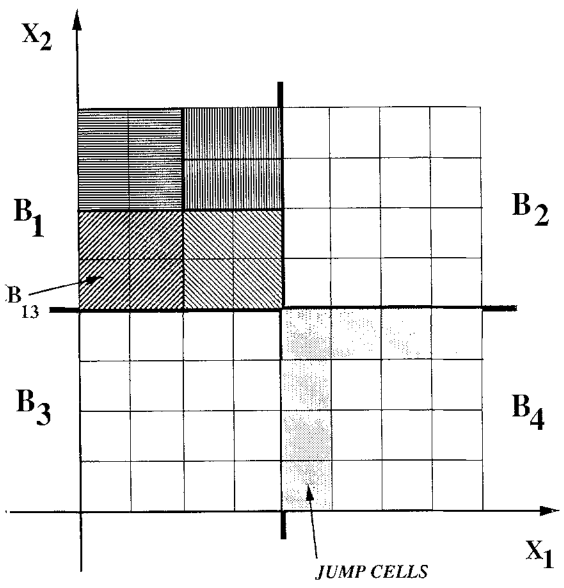

Fig. 3. Illustration of partitioning of state space for recursive unraveling of the adjoining cell mapping.

level of analysis). After performing the second level of analysis, the remaining active cells are grouped into the corresponding blocks B_1 to B_4 (of the first partition level) to perform the first level of analysis. Then, only the remaining active cells need to be considered for the analysis of the entire region. This displays the recursive nature of the partition and analysis procedures. The reduction of memory requirements with respect to the simple cell mapping method is obvious. For instance note that, after analyzing block B_4, the remaining active cells from such a block which must be considered for the analysis of the entire region form a subset of the jump cells as shown in Figure 3.

4.3. Practical Implementation of Recursive Unraveling Algorithm

We focus now on the details of practical implementation of the recursive unraveling algorithm. The implementation makes use of the following observations:

1. First define different levels of partition. Then, start the analysis of the (smaller) blocks corresponding to the first (lower) level of analysis.
2. Since an adjoining mapping is preserved in each block it guarantees that
   (i) Every transition cell maps to either the sink cell or another cell in the same block.
   (ii) Only jump cell can map to a cell in another block.
3. An unraveling algorithm is applied to the adjoining mapping within the block. The algorithm provides the following information:
   (i) Every set of periodic cells with all its cells belonging to a unique block is detected.
   (ii) Every jump cell can be classified into one of the three possible categories:
      (a) Jump cells that map to a periodic solution.
      (b) Jump cells that finally map to another jump cell (maybe through some transition cells).

(c) Jump cells that map outside the current block being analyzed. The information regarding where they map will be determined at a posterior level of analysis.

(iii) After finishing the analysis of a block, all its transition cells and jump cells which map to a periodic solution are classified as inactive cells. We do not need to keep the information regarding where such cells map.

4. The active cells are kept for further analysis in a higher (posterior) analysis level. This analysis level corresponds to a higher (previous) partition level.
5. The next analysis level consists of regrouping two or more already processed blocks into a new block. The following points are to be noted in this step:
   (i) Only the active cells of the previous analysis level are considered.
   (ii) In general, some of the cells that were jump cells in the previous level of analysis become transition cells in the new block.
   (iii) Considering the new block as a whole, we compute again the adjoining mapping for those cells which mapped to an outside block in the previous level of analysis. These cells may map to a cell that belongs now to the block considered in this higher level of analysis and some of them may still map to a cell that is outside this block.
   (iv) The mappings of cells in the new block are defined only among the active cells of the previous level of analysis. The analysis of periodic cells is similar to the one performed in the previous level of analysis.
6. The algorithm terminates when the higher and last level of analysis is performed over the block corresponding to the total state space under analysis.

4.4. Additional Remarks

We note that when a periodic solution is detected at a given level of analysis, it may not be composed of adjoining cells since it is only characterized by the corresponding active cells. Nevertheless, once a periodic solution is detected, it is straightforward to "complete" the whole set of adjoining cells which make up the solution. For instance, one can choose any one of the detected cells and calculate its adjoining mapping. The corresponding adjoining cell must also belong to this periodic solution and one can repeatedly apply the same procedure to each adjoining cell so obtained. By following the adjoining mappings, one can successively locate all the cells that belong to the detected periodic solution where each cell is adjoining to one another.

The specific recursive procedure developed and implemented while performing computations in this paper is based on successive partitioning of each block into 2^N new blocks. This is done by partitioning the blocks along every state space variable into two parts, each with half the number of intervals in every variable. An appropriate initial first partition level of the entire space prepares the blocks of the second level to meet the required conditions (2^N intervals in each state variable). The partitioning of blocks in the algorithm is terminated when a prescribed small block size limit is reached in a corresponding partition level (in our implementation, blocks of size $2 \times 2 \times \ldots \times 2$).

After defining the partition, one is concerned with the management of memory needed to define the mapping of active cells. The active cells at a given level of analysis are a subset of the jump cells. The percentage of active cells among the jump cells is problem dependent.

One may implement the recursive unraveling algorithm in such a way that the mapping of the jump cells is always defined. Then for the number of state variables $N > 1$ and for the partition defined above, the maximum number of mappings to be stored (i.e., the number of jump cells), denoted by M_1, is given by

$$M_1 = \prod_{i=i}^{N} n_i - 2^N \prod_{i=1}^{N} \left(\frac{n_i}{2} - 1\right),$$

where n_i is the number of intervals along the state variable x_i. On the other hand, if many of the jump cells become inactive, one may define which cells are the active ones and store their mappings. The total memory M_2 needed for this purpose is given by

$$M_2 = 2EM_1\,, \tag{13}$$

where E is the ratio of the active cells to the jump cells and depends on the dynamics of the system under study.

An implementation of the recursive unraveling algorithm in this paper corresponds to the second type where the memory requirements depend on the parameter E. Finally, note that when $N = 1$, no limit cycles are possible and the location of invariant sets is trivial, requiring the storage of two mappings only.

5. Comparative Properties

When compared with the simple cell mapping method, the new procedure of adjoining cell mapping developed above has several advantages which are described below.

5.1. Features of Adjoining Mapping

1. The problem of how to choose the time of integration for cell mapping analysis is answered. Only a starting value of integration time T needs to be provided. Furthermore, this initial guess can be determined in an adaptive way. Therefore, the influence of T is minor in the final results as noted in [9].
2. The adjoining mapping works very well when the vector field **F** takes on large values (regions of "fast" trajectories). For instance, the mapping follows the paths very clearly providing information regarding the flow of trajectories in the region under study. In such situations, the simple cell mapping method generates big jumps making it difficult to interpret the computational results, especially the global domains of attraction.
3. When detecting limit cycles, a simple cell mapping may replace a limit cycle by more than one periodic group of cells, which are disconnected and highly dependent on the duration of integration. An adjoining mapping must provide periodic groups of adjoining (connected) cells. This property concords with the continuous nature of the limit cycle in state space.
4. When applying the simple cell mapping method to systems of the form analyzed in [1, 6, 7, 8], problems may arise when the vector field **F** does not evolve smoothly. It has been pointed out there that major problems appear while studying the evolution of trajectories that are close to the singular manifolds. The trajectories in these regions tend to evolve very fast and the traditional computational techniques do not work. Since the simple cell mapping algorithm integrates trajectories starting in each cell over a fixed duration of time when calculating its mapping, systems mentioned above may generate very big "jumps" in the mapping. Hence, it

is difficult to interpret global domains of attraction for this type of systems. The adjoining cell mapping method avoids this problem by automatically adjusting the integration duration to be small whenever needed so that the mapping is always from a cell to an adjoining cell. Hence, the adjoining cell mapping technique can provide invaluable information regarding the evolution of trajectories in these critical regions. This aspect will be clearly illustrated with examples provided in the sequel of this paper (see [9]).

5. Since a cell can only map to a neighboring one, any required memory storage procedure is highly simplified. Such a simplification can be generally extended to other refinements of the simple cell mapping method (for example, the generalized cell mapping technique, see [4]).

5.2. Features of Recursive Unraveling Algorithm

1. The memory required at each level of partition is the same as that for the active cells in the previous level of analysis. These active cells are a subset of the jump cells. In addition, the jump cells are a subset of the boundary cells of the blocks in the previous analysis level. Since the boundary of a set has one dimension less than the set, the required memory is reduced accordingly.
2. A proper choice of the partition (especially at the first level) can make the boundaries of the blocks to be defined along the state space variables requiring less number of cells. Therefore, a reduction in memory by a factor of $n_{\max}/2$ can be accomplished ($n_{\max}$ is the maximum number of cell divisions required along a state space variable).
3. Additional computation required by the recursive unraveling algorithm is primarily based on the iterative technique used to determine the adjoining cell mapping associated with each cell. The fact that the mapping of some cells may have to be computed more than once does not prove to be critical, because this will only happen for a few jump cells.
4. A recursive procedure is very easy to implement thus making the adjoining cell mapping algorithm very compact in this case.
5. The structure of the algorithm is ideally suited for parallel processing implementations.
6. This approach can be very useful for future development of adaptive and recursive schemes for the determination of global regions of attraction.

6. Concluding Remarks

A new method of cell mapping, based on an adaptive integration scheme, is introduced in this paper. The cell mapping approach developed here guarantees the existence of an adjoining cell mapping and allows recursive development of algorithms for determining the global behavior of nonlinear dynamical systems. The adaptive mapping unraveling algorithm presented here overcomes the problem of selection of an appropriate integration time required to create an associated cell mapping for an autonomous system.

A recursive unraveling algorithm for adjoining mapping is also presented for locating equilibrium and periodic solutions of the system. Incidentally, a practical implementation of the unraveling algorithm reduces memory requirement due to its recursive nature. The memory requirement is problem dependent, however, it reduces by a factor $n_{\max}/2$ when compared with the simple cell mapping method (where $n_{\max}$ is the largest number of divisions performed in any state variable).

The method of adjoining cell mapping can locate limit cycles very effectively as well as unravel pathological dynamics. In the same spirit, the adjoining cell mapping method is applicable to the generalized cell mapping formulation of autonomous dynamical systems.

Acknowledgements

The work reported here is partially supported by a grant from the National Science Foundation and the USC Faculty Research and Innovation Fund. P. J. Zufiria also thanks the support of a Formación de Postgrado fellowship of the Programa Nacional de F.P.I. provided by the D.G.I.C.T. of the Ministerio de Educación y Ciencia of Spain.

References

1. Guttalu, R. S. and P. J. Zufiria, 'On a class of nonstandard dynamical systems: Singularity issues', in C. T. Leondes (ed.), *Control and Dynamic Systems* **34**, 1990, 279–324.
2. Hsu, C. S., 'A theory of cell-to-cell mapping dynamical systems', *Journal of Applied Mechanics* **47**, 1980, 931–939.
3. Hsu, C. S. and R. S. Guttalu, 'An unravelling algorithm for global analysis of dynamical systems: An application of cell-to-cell mappings', *Journal of Applied Mechanics* **47**, 1980, 940–948.
4. Hsu, C. S., *Cell-to-Cell Mapping: A Method of Global Analysis for Nonlinear Systems*, Springer-Verlag, New York, 1987.
5. Larson, R. E., *State Increment Dynamic Programming*, American Elsevier Publishing Company, New York, 1968.
6. Zufiria P. J., 'Global behavior of a class of nonlinear dynamical systems: Analtyical, computational and control aspects', Ph.D. Dissertation, University of Southern California, 1989.
7. Zufiria, P. J. and R. S. Guttalu, 'On an application of dynamical system theory to determine all the zeros of a vector function', *Journal of Mathematical Analysis and Applications* **152**, 1990, 269–295.
8. Zufiria, P. J. and R. S. Guttalu, 'A computational method for locating all the roots of a vector function', *Applied Mathematics and Computation* **35**, 1990, 13–59.
9. Guttalu, R. S. and P. J. Zufiria, 'The adjoining cell mapping and its recursive unraveling, Part II: Application to selected problems', *Nonlinear Dynamics* **4**, 1993 (next issue).