ADAPTIVE SET-ORIENTED COMPUTATION OF TOPOLOGICAL HORSESHOE FACTORS IN AREA AND VOLUME PRESERVING MAPS

J.D. MIRELES JAMES

Abstract. We describe an automatic chaos verification scheme based on set oriented numerical methods, which is especially well suited to the study of area and volume preserving diffeomorphisms. The novel feature of the scheme is an iterative algorithm for approximating connecting orbits between collections of hyperbolic fixed and periodic points with greater and greater accuracy. The algorithm is geometric rather than graph theoretic in nature and, unlike existing methods, does not require the computation of chain recurrent sets. We give several example computations in dimension two and three.

Key words. set-oriented numerics, conservative dynamics, automatic chaos verification, connecting orbits

AMS subject classifications. 65G20,65-04, 65P10, 65P20, 37B30, 37B10, 37C05, 37C29, 37C50

1. Introduction.

The main goal of this paper is to develop a set of computational tools, based on set-oriented numerical methods, which automatically, efficiently, and rigorously verify the existence of chaotic subsystems of conservative dynamical systems. Unlike the dissipative case, we cannot exploit phase space contraction in the design of efficient algorithms. Instead, we fix *a-priori* a particular geometric mechanism called a tangle, and develop algorithms which converge iteratively to chaotic subsystems associated with the tangle. We also develop an interface between our algorithms and tools from the discrete Conley Index theory, in order to *a-posteriori* validate the results of our computations. We illustrate the effectiveness of these tools in several example computations for area and volume preserving diffeomorphisms.

Our computations are built on the so called set-oriented numerical methods, which we discuss in Sections 2.1 and 2.2. Loosely speaking, a set-oriented discretization of a dynamical system consists of discretizing the domain of the system by a finite grid of cubes, and discretizing the dynamics by associating with each cube in the domain a collection of grid cubes which cover its image. With set-oriented methods, one studies the evolution of sets under combinatorial dynamics, rather than following iterates of individual points.

Set-oriented methods have been applied to a wide variety of computational problems in dynamical systems, including orbit design in celestial mechanics [DJK$^+$05, DJPT06], optimal control theory [GJ05], bifurcations giving rise to connecting orbits [DJT01], computation of invariant measures on attractors [Jun01], computation of global dynamics over a wide range of parameters in ecological models [AKK$^+$09], and automatic chaos verification [DJM05, DFT08]. Expositions of set-oriented methods are found, for example, in [Osi07, KMM04, GJ05].

A typical set-oriented computation consists of two parts. In the first part, qualitative algorithms are used to locate some interesting combinatorial dynamics. The second part is validation, where one attempts to rigorously establish that some properties of the combinatorial dynamics are inherited by the original system. The validation step is necessary as the combinatorial discretization of the dynamical system always involves some loss of information.

When working with dissipative systems, it is natural to exploit the Conley Decomposition Theorem [HMZ88] during the qualitative portion of the computation. This far reaching generalization of Morse decompositions states that the phase space of a dissipative dynamical system is decomposed into chain recurrent components, and gradient-like sets. This approach leads to a very complete theory of set-oriented computations for dissipative systems. For details see [KMV05, HB06].

Another powerful qualitative tool is Smale's Tangle Theorem [Sma65], which states that the transverse crossing of the stable and unstable manifolds of a hyperbolic fixed point gives rise to a chaotic subsystems called a tangle. In [BW95], this theorem is weakened to require only topological crossing of the stable and unstable manifolds, and extended to collections of hyperbolic fixed and periodic points. Since crossings of the stable and unstable manifolds of fixed points give rise to heteroclinic or homoclinic orbits, tangles are often detected and localized in set-oriented computations by searching for combinatorial connecting orbits between fixed and periodic points.

The following set-oriented meta-algorithm sketches the proof of the existence of a topological horseshoe factor in a dissipative dynamical system system (X, f). (Topological horse shoe factors are defined in Section 2.3). The meta-algorithm is in the spirt of [DJM05, DFT08].

Algorithm 1.1 (Chaos Proof Meta-Algorithm; Dissipative Case).

```
function chaosSearch ($X_0, f$)
Part 1: (Qualitative Computation)
    Let $X = X_0$
    Let $(\mathcal{X}, \mathcal{F})$ = discretization of $(X, f)$
    do (until resolution of $\mathcal{X}$ is fine enough)
        Compute $\mathcal{S}$ = Invariant Part of $\mathcal{X}$
        Subdivide $\mathcal{S}$
        Let $(\mathcal{X}, \mathcal{F})$ = discretization of $(\mathcal{S}, f)$
    end

    • Choose some collection of combinatorial fixed or periodic orbits in $\mathcal{X}$.
    • Compute combinatorial connecting orbits.
Part 2: (Verification)
    • Grow an index pair for the special orbits and the connections.
    • Compute discrete Conley Index index of the pair.
Return the index pair and the Conley Index.
End Algorithm
```

Remarks 1.2.

1. *The discretization and invariant part computation are discussed in Sections 2.1 and 2.2 respectively. We note that the invariant part computation is combinatorial, and produces a cover of the chain recurrent set of $f|_X$ by cubes in $\mathcal{X}$. By subdividing and recomputing the discretization, the iterative step of the algorithm produces a decreasing sequence of finer and finer approximations to the chain recurrent set.*
2. *Once the chain recurrent set has been localized at high resolution, some connecting orbits are typically computed using graph theoretic shortest paths algorithms such as Dijkstra's Algorithm. (A reference on graph algorithms we found useful is [MMP05]).*
3. *The verification stage, is discussed in Section 2.3. There, we recall the formal definitions of index pair and the discrete Conley Index. Studying the homo-*

morphism induced by f on the relative homology of the Index Pair leads to the proof that f has a chaotic subsystem in X_0. (See the worked examples in [KMM04], Chapter 10.7*)*

4. *In terms of the present work, the essential point concerning Algorithm 1.1 is that since the chain recurrent components of a dissipative dynamical systems have measure zero, the iterative procedure in Algorithm 1.1 can produce major reductions in the region of phase space which needs to be searched. Expensive topological and graph theoretic computations need only be carried out on this greatly reduced region.*

Conservative systems admit invariant sets of large measure, so that the iterative loop in Algorithm 1.1 is of little or no help. (See Example A.1 in the Appendix). We make the following modification to the meta-algorithm; instead of computing global chain recurrent sets in the qualitative step, we choose *a-priori* some hyperbolic fixed or periodic orbits, and apply an iterative geometric algorithm which localizes combinatorial connecting orbits directly. Here is a sketch of our scheme;

ALGORITHM 1.3 (Chaos Proof Meta-Algorithm; Conservative Case).

```
Fix $p$; some hyperbolic fixed or periodic points.
function conservativeChaosSearch $(p, X, f)$
Part 1: (Qualitative Computation)
        Let $(\mathcal{X}, \mathcal{F})$ = discretization of $(X, f)$
        do (until resolution of $\mathcal{X}$ is high enough)
                $\mathcal{S}$ = reliable combinatorial connecting orbits for $p$.
                Subdivide $\mathcal{S}$
                Let $(\mathcal{X}, \mathcal{F})$ = discretization of $(\mathcal{S}, f)$
        end
Part 2: (Verification)
                ⋮
return
```

The main result of the paper is Algorithm 3.7 in Section 3.1. Algorithm 3.7 computes $\mathcal{S}$ in the iterative loop of Algorithm 1.3, for the case when p is a hyperbolic fixed point. In Section 4 we look at procedures for some other choices of p.

REMARKS 1.4.

1. *In Examples A.2 and A.3 in the Appendix, we look at difficulties which arise if we naively apply graph theoretic shortest path algorithms to compute $\mathcal{S}$ in the iterative loop. The material in the appendix motivates the need for algorithm 3.7.*
2. *Algorithm 3.7 is not strictly graph theoretic. Rather, the algorithm is based on insights provided by the qualitative theory of dynamical systems. Namely Smale's Tangle Theorem [Sma65] and the λ-Lemma (see [PdM82]).*
3. *By focusing on zero dimensional objects such as tangles, we are able to rapidly cull the phase space for conservative systems in much the same way as is done for dissipative systems in Algorithm 1.1.*
4. *Algorithm 1.3 is in the spirit of the quantitative theory of Hamiltonian systems developed in [MG08], [GR04], [GdlL06], [RdlL], and [Rob02]. In fact the present work can be seen as an attempt to blend the qualitative work just mentioned with the computational theory of [KMV05], [HB06], in order to bring conservative systems into the automated chaos verification paradigm of [DJM04a], and [DJM05].*

5. *Strictly speaking, Algorithm 1.3 does not require f to be conservative. In fact, even though a satisfactory theory of automatic chaos verification in dissipative system already exists, it is possible that in higher dimensions the cost of computing global chain recurrent sets could impose the same constraints required by conservative systems. Namely a-priori focus on some interesting set p and application of a scheme like Algorithm 1.3.*

The remainder of the paper is organized as follows. Sections 2.1 and 2.2 review the formal definitions of set-oriented discretization and combinatorial dynamics, as well as the notation and fundamental algorithms of set oriented numerics. Section 2.3 reviews some notions from the discrete Conley Index theory, which we use in the validation stage of the computations.

In Section 3.1 we present Algorithm 3.7. Section 4 presents four example computations. Each example highlights some important feature of our methods.

In Example 4.1 we use meta-algorithm 1.3 in conjunction with Algorithm 3.7, and the Conley Index methods discussed in Section 2.3 to verify the existence of a topological horseshoe factor for a non-perturbative parameter value in the standard map.

In Example 4.2 we use a modification of Algorithm 3.7 to compute combinatorial connections between the two hyperbolic fixed points of the Suris map. Example 4.2 shows that the methods of computation used on Example 4.1 are easily modified to compute heteroclinic tangles. Examples 4.1 and 4.2 illustrate clearly the fact that our scheme is able to automatically culls invariant tori form the computations, and avoid the saturation with undesired cubes exhibited in Appendix Example A.1.

A third and final planar example is Example 4.3. Here we discuss the computation, in the area preserving Henon map, of a heteroclinic tangle with a different geometry than in Example 4.2. The difference in geometry poses no problems for our automated scheme. In addition, we choose parameters for the map close to the 'anti-integrable limit', where no phase space structure is easily seen using classical phase space sampling. This highlights that our scheme can automatically cull tori, but it in no way depends on their presence.

The final computation is Example 4.4, where we compute a homoclinic tangle for the volume preserving ABC map. Example 4.4 shows that the utility of the ideas presented in this paper are not limited to the plane.

In Section 5 we discuss the implementation of our software, and its performance. Implementation is discussed in some detail, so as to give the performance notes some context. We also list specific files in which to find the source code if the implementations. Finally, in the appendix we give specific numerical examples which highlight some difficulties that arise when applying set-oriented computations to conservative systems.

2. Background and Notation.

In this section we review material from [KMV05], [DJM05], [BW95], [KMM04], [HMZ88], [HB06], [Srz00], [MM98], [DH97a], [Szy95], and [Szy97], which we use throughout the remainder of the paper.

2.1. Combinatorial Representation of a Dynamical Systems.

Let $M \subset \mathbb{R}^N$ and $f : M \to M$ be a homeomorphism. The pair (M, f) is a discrete time topological dynamical system. In order to study a dynamical system using the computer, it is necessary to discretize both M and f, meaning that we must choose some finite representation of each. To this end, we make the following geometric definitions. The first set of definitions explains the discretization of phase space, the second explains the discretization of the dynamics.

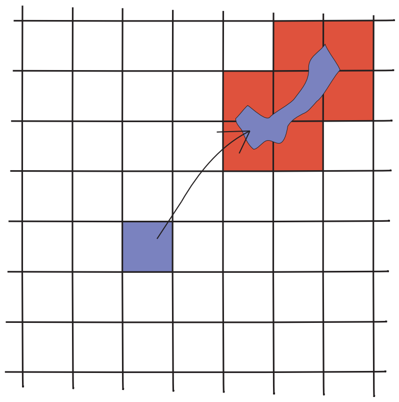

Fig. 2.1. *A cube Q and its image under f. The combinatorial image $\mathcal{F}(Q)$ is a set of grid cubes which contain $f(Q)$ on its interior. The combinatorial image pictured here is minimal, but this need not always be the case.*

Definition 2.1. *(Set-Oriented Discretization of Phase Space)*

1. *A full cube Q in $\mathbb{R}^n$ is a product of intervals $Q = [a_1, b_1] \times \ldots \times [a_n, b_n]$ where $a_i < b_i$ for all $1 \leq i \leq n$. A cube $Q \subset \mathbb{R}^n$ is a uniform cube if $b_i - a_i = r > 0$ for some fixed $r \in \mathbb{R}$ and all $1 \leq i \leq n$. We call r the resolution of the uniform cube.*
2. *Let $\mathbb{Z}^n$ be the set of vectors with integer coordinates, and fix both an $\alpha \in \mathbb{R}^n$, and an $0 < r \in \mathbb{R}$. The uniform cubical grid with resolution r and origin α, denoted $\mathcal{G}_{r,\alpha}$ is defined to be the set*

$$\mathcal{G}_{r,\alpha} = r \cdot \mathbb{Z}^n + \alpha.$$

   The r and α subscripts are suppressed for simplicity whenever there is no danger of confusion. The elements of $\mathcal{G}$ are called grid points.
3. *If $q \in \mathbb{Z}^n$ we say that $q = (q_1, \ldots, q_n)$ are the grid coordinates of the grid point $v = r \cdot q + \alpha \in \mathcal{G}_{r,\alpha}$. The unit cube based at q in $\mathbb{Z}^n$ is the cube $[q_1, q_1 + 1] \times \ldots \times [q_n, q_n + 1]$, which we denote B_q.*
4. *The grid cube with base $v = rq + \alpha$ in $\mathcal{G}$ is defined to be*

$$Q_v = r \cdot B_q + \alpha.$$

   Note that Q_v is a full uniform cube in $\mathbb{R}^n$ with resolution r. To say simply that Q is a grid cube of $\mathcal{G}$ means that Q is a grid cube with base v for some unspecified $v \in \mathcal{G}$.
5. *A finite collection $\mathcal{X}$, of grid cubes in $\mathcal{G}$, is called a uniform full cubical complex in $\mathbb{R}^n$ subordinate to the grid $\mathcal{G}_{r,\alpha}$. A cubical complex is a topological space under the subspace topology inherited from $\mathbb{R}^n$. In the remainder of this paper a cubical complex is always a full uniform cubical complex subordinate to some grid $\mathcal{G}$.*
6. *If X is compact subset of $\mathbb{R}^n$, an outer cubical cover of X is a cubical complex $\mathcal{X}$ such that $X \subset \mathcal{X}^\circ$. Throughout this paper, a cubical cover always means an outer cubical cover.*
7. *Suppose $\mathcal{X}$ is a cubical cover of a compact set $X \subset \mathbb{R}^n$ such that for each $Q \in \mathcal{X}$, $Q \cap X \neq \emptyset$. We say that $\mathcal{X}$ is a minimal cubical cover of X.*

Definition 2.2. *(Set-Oriented Discretization of Dynamics)*

1. *A combinatorial image, $\mathcal{F}(Q)$ of $Q \in \mathcal{X}$, is defined to be an outer cubical covering of $f(Q)$ by cubes of $\mathcal{X}$. More precisely let $\{Q_j\}_{j\in I}$ be any collection of cubes in $\mathcal{X}$ having*

$$f(Q) \subset \left(\bigcup_{j\in I} Q_j \right)^{\circ},$$

and take

$$\mathcal{F}(Q) = \bigcup_{j\in I} Q_j.$$

We refer to the condition $f(Q) \subset \mathcal{F}(Q)^{\circ}$ as the outer enclosure property of $\mathcal{F}$. The definition is due to [Szy97]. The term symbolic image is used by [Osi07]. A schematic of this construction is given in Fig 2.1.

2. *If, in addition $\mathcal{F}(Q)$ is a minimal cubical covering of $f(Q)$, then we call $\mathcal{F}(Q)$ the minimal combinatorial image of Q. (Minimality depends on a fixed choice of underlying grid $\mathcal{G}_{r,\alpha}$).*
3. *The association of each $Q \in \mathcal{X}$ a combinatorial image $\mathcal{F}(Q)$ defines a combinatorial outer enclosure of f on $\mathcal{X}$. The enclosure is a function $\mathcal{F} : \mathcal{X} \to$ **Pow**$(\mathcal{X})$ (where **Pow** is the collection of all subsets of $\mathcal{X}$). In order to highlight the fact that this is a multivalued function, the notation*

$$\mathcal{F} : \mathcal{X} \rightrightarrows \mathcal{X}$$

is used. $\mathcal{F}$ is also refereed to as a combinatorial multivalued map. In this paper, combinatorial enclosure always means an outer enclosure.

4. *We call the pair $(\mathcal{X}, \mathcal{F})$ a combinatorial dynamical system. This is the notion of discretization we use throughout the paper.*
5. *The combinatorial inverse $\mathcal{F}^{-1}$ of $\mathcal{F}$ is defined to be*

$$\mathcal{F}^{-1}(Q_i) = \{Q_j \in \mathcal{X} \ : \ Q_i \in \mathcal{F}(Q_j)\}.$$

Note that the combinatorial inverse $\mathcal{F}^{-1}$ also has the outer enclosure property, in the sense that

$$f^{-1}(Q) \subset \left(\mathcal{F}^{-1}(Q)\right)^{\circ}$$

6. *Let $\mathcal{A} = \bigcup_{i\in I} Q_i$ be a cubical subset of $\mathcal{X}$. Define the combinatorial image of $\mathcal{A}$ by $\mathcal{F}(\mathcal{A}) = \bigcup_{i\in I} \mathcal{F}(Q_i)$. Similarly, define $\mathcal{F}^n(\mathcal{A})$ to be the set defined inductively by $\mathcal{F}^0(\mathcal{A}) = \mathcal{A}$ and $\mathcal{F}^n(\mathcal{A}) = \mathcal{F}^{n-1}(\mathcal{A})$ for all integer $n \geq 1$. Define $\mathcal{F}^{-n}(\mathcal{A}) = [\mathcal{F}^{-1}]^n(\mathcal{A})$*
7. *The dynamical graph associated with $(\mathcal{X}, \mathcal{F})$, denoted $\mathfrak{G}_{\mathcal{X},\mathcal{F}}$, is the directed graph wherein we associate a node with each $Q \in \mathcal{X}$, and a directed arrow from node Q to node Q' if and only if $Q' \in \mathcal{F}(Q)$. We suppress the subscripts when the combinatorial dynamical system is understood. The dynamical graph $\mathfrak{G}^{-1}$ associated with $\mathcal{F}^{-1}$ is obtained by reversing the direction of the arrows of $\mathfrak{G}$.*
8. *For $Q \in \mathcal{X}$, a combinatorial trajectory through Q under $\mathcal{F}$ is a function $T_Q : \mathbb{Z} \to \mathcal{X}$ so that $T_Q(0) = Q$ and $T_Q(n+1) \in \mathcal{F}(T_Q(n))$ for all $n \in \mathbb{Z}$.*

9. *An combinatorial orbit segment $T_Q|_I$ is the restriction of an orbit T_Q to a finite set of integers of the form $I = \{0, \ldots, n\}$. In terms of the dynamical graph, a combinatorial orbit segment $T_Q|_I$ is path through Q.*
10. *Let $Q \in \mathcal{X}$, and T_Q be a combinatorial trajectory through Q under $\mathcal{F}$. T_Q is called a false positive if there exist $n \in \mathbb{N}$, and $Q' \in \mathcal{X}$ so that $Q' = T_Q(n)$ but $f^n(x) \notin Q'$ for all $x \in Q$.*
11. *Let $\mathcal{N} \subset \mathcal{X}$. The maximal combinatorial invariant subset of $\mathcal{N}$ under $\mathcal{F}$ is the set*

$$Inv(\mathcal{N}, \mathcal{F}) = \{Q \in \mathcal{N} \ : \ \exists\, T_Q : \mathbb{Z} \to \mathcal{N} \subset \mathcal{X}\}.$$

12. *Suppose that $\mathcal{S} = Inv(\mathcal{N}, \mathcal{F})$ for $\mathcal{N} \subset \mathcal{X}$ and $\mathcal{S} \neq \emptyset$. The sets*

$$\mathcal{W}^s_{\mathcal{N}}(\mathcal{S}) = \{Q \in \mathcal{N} \ : \ \textit{there is a } T_Q : \mathbb{Z} \to \mathcal{N} \textit{ so that}$$

$$T_Q(n) \in S \textit{ for some } n \geq 0\},$$

and

$$\mathcal{W}^u_{\mathcal{N}}(\mathcal{S}) = \{Q \in \mathcal{N} \ : \ \textit{there is a } T_Q : \mathbb{Z} \to \mathcal{N} \textit{ so that}$$

$$T_Q(n) \in S \textit{ for some } n \leq 0\},$$

are called the combinatorial stable and unstable subsets of $\mathcal{S}$, relative to $\mathcal{N}$.
13. *Let $\mathcal{A} \subset \mathcal{X}$. Then*

$$\mathcal{C}_k = \bigcup_{i=0}^{k} \mathcal{F}^k(\mathcal{A}),$$

with $k \in \mathbb{Z}$, is called a k-set orbit of $\mathcal{A}$. If $k > 0$ then $\mathcal{C}_k$ is called a forward orbit, while if $k < 0$, $\mathcal{C}_k$ is called a backward or inverse orbit. C_k is also refereed to as a globalization of $\mathcal{A}$.

Remarks 2.3.

(a) *It is perfectly acceptable to have $F(Q) = \emptyset$ for some $Q \in \mathcal{X}$. This often happens in applications when $f : M \to M$ is a homeomorphism which we discretize on some compact sub-domain $X \subset M$, with X not invariant under f. On the graph level, Q corresponds to a node with no outgoing arrow.*

(b) *If $\mathcal{G}_{r,\alpha}$ is a fixed cubical grid, X is a fixed compact set, and $\mathcal{X}$ is the minimal cubical covering of X relative to $\mathcal{G}$, then the function $f : X \to \mathbb{R}^n$ has a unique minimal combinatorial outer enclosure. In practice it is difficult to compute the minimal combinatorial outer enclosure and we settle for computing a combinatorial outer enclosure. Our implementation of this computation is explained in detail in Section 5*

(c) *Images of cubical subsets inherit the outer enclosure property of individual cubes, as*

$$f(\mathcal{A}) = f\left(\bigcup_{i\in I} Q_i\right) = \bigcup_{i\in I} f(Q) \subset \bigcup_{i\in I} [\mathcal{F}(Q)]^\circ \tag{2.1}$$

$$= \left[\bigcup_{i\in I} \mathcal{F}(Q)\right]^\circ = [\mathcal{F}(\mathcal{A})]^\circ.$$

(d) *We make the following definition: Suppose that for some $Q \in \mathcal{X}$, and $x \in Q$ there is a $Q' \in \mathcal{X}$ so that $f^n(x) \in Q'$, but that $\mathcal{F}$ does not admit combinatorial trajectory through Q with $T_Q(n) = Q'$. In this case we say that $\mathcal{F}$ admits a false negative. Note that the outer enclosure property prohibits false negatives.*

(e) *On the level of graph dynamics, $S = Inv(\mathcal{N}, \mathcal{F})$ is the strongly connected component of the the subgraph of $\mathfrak{G}_{\mathcal{X}}$ associated with $\mathcal{N}$.*

(f) *The existence of a nonempty combinatorial invariant subset does not imply the existence of a pointwise f-invariant subset $S \subset \mathcal{S}$. Nevertheless, we can claim that if $\mathcal{N} \subset \mathcal{X}$ and $Inv(\mathcal{N}, \mathcal{F}) = \emptyset$, then the pointwise maximal f invariant subset in $\mathcal{N}$ is empty, as an outer enclosures does not admit false negatives.*

(g) *Q is in the local stable set of $\mathcal{S}$ relative to $\mathcal{N}$ if there is a combinatorial trajectory T_Q, and a $Q' \in \mathcal{S}$ so that for some $n \geq 0$, $T_Q(n) = Q'$ and so that $T_Q(i) \in \mathcal{N}$ for $0 \leq i \leq n$. A similar comment holds for the unstable set by considering $\mathcal{F}^{-1}$. Since, once a trajectory enters $\mathcal{S}$ it can stay there for all time, the unstable and stable sets are made up of cubes for which there are combinatorial orbits which stay in $\mathcal{N}$ for all backward or respectively forward time.*

2.2. Set Oriented Numerical Algorithms. This section presents the core set-oriented algorithms used in the remainder of the paper. First we collect several simple utility functions which we use freely in the sequel. The functions below depend on a cubical complex $\mathcal{X}$, the underlying grid $\mathcal{G}_{r,\alpha}$, and the cubical multivalued map $\mathcal{F}$, even when this dependence is not explicitly indicated.

1. $\#(\mathcal{X})$ returns the integer number of cubes in $\mathcal{X}$. Recall that this is finite by definition.
2. **dimensionOf**$(\mathcal{X})$ returns the dimension $\mathcal{X}$,
3. **resolutionOf**$(\mathcal{X})$ returns the resolution of $\mathcal{X}$.
4. Let $\mathcal{N} \subset \mathcal{X}$ a cubical subset. The function
$$\mathbf{wrap}_{\mathcal{X}}(\mathcal{N}) = \{Q \in \mathcal{X} \ : \ Q \cap \mathcal{N} \neq \emptyset\}$$
returns $\mathcal{N}$ and all of its neighbors in $\mathcal{X}$.
5. For $\mathcal{N} \subset \mathcal{X}$, denote the restriction of $\mathcal{F}$ to $\mathcal{N}$ by
$$\mathcal{F}' = \mathbf{restrict}(\mathcal{F}, \mathcal{N}),$$
where $\mathcal{F}' : \mathcal{N} \rightrightarrows \mathcal{N}$ and $\mathcal{F}'(Q) = \mathcal{F}(Q) \cap \mathcal{N}$, for all $Q \in \mathcal{N}$.
6. **collar**$(\mathcal{N}) =$ **wrap**$(\mathcal{N}) \backslash \mathcal{N}$.
7. For a cubical complex $\mathcal{X}$ subordinate to the grid $\mathcal{G}_{r,\alpha}$, let
$$\mathcal{X}' = \mathbf{subdivide}(\mathcal{X})$$
be the cubical complex obtained by including $\mathcal{X}$ into the refined cubical grid $\mathcal{G}_{r/2,\alpha}$. Note that
$$\#(\mathbf{subdivide}(\mathcal{X})) = 2^n \, \#(\mathcal{X}),$$
where $n =$ **dimensionOf**$(\mathcal{X})$.
8. For a dynamical system (M, f), let
$$(\mathcal{X}, \mathcal{F}) = \mathbf{enclose}(\mathcal{X}, f)$$
denote the discretization operator which returns the combinatorial dynamical system $(\mathcal{X}, \mathcal{F})$. The implementation of this operator is discussed in Section 5.

Algorithm 2.5 below computes the maximal combinatorial invariant subset of a cubical subset as defined in section 2.1. We also explain how the invariant part algorithm can be modified to compute stable and unstable subsets. The algorithms described in this section are the fundamental building blocks of the set-oriented numerical methods we develop in the remainder of the paper. For discussion of the convergence and complexity of the algorithms, see [DH96], [KMM04], [DH97b], [Osi07], and [DH97a]. We begin with a theorem.

THEOREM 2.4 (Combinatorial Invariant Sets). *Let $\mathcal{X}$ be a cubical cover of a compact $X \subset \mathbb{R}^n$, and let $\mathcal{F} : \mathcal{X} \to \mathcal{X}$ be an outer combinatorial enclosure for a homeomorphism $f : X \to X$. Suppose that $\mathcal{N} \subset \mathcal{X}$, and define the sequence*

$$\mathcal{S}_0 = \mathcal{N}$$

$$\mathcal{S}_{j+1} = \mathcal{F}(\mathcal{S}_j) \cap \mathcal{F}^{-1}(\mathcal{S}_j) \cap \mathcal{S}_j$$

Then there is a $K \in \mathbb{N}$ so that

$$\mathcal{S}_K = \mathcal{S}_{K-1},$$

and we have that

$$\mathcal{S}_K = Inv(\mathcal{N}, \mathcal{F}).$$

For the proof, see [KMM04] section 10.6.

Note that neither $\mathcal{S}_K = \emptyset$ nor $\mathcal{S}_K = \mathcal{N}$ are ruled out, and that if the sequence is constant at one step then it is constant at all future iterations. Similar theorems can be stated and proved for the local stable and unstable sets $\mathcal{W}^{s,u}_{\mathcal{N}}(\mathcal{S})$ of the combinatorial invariant set $\mathcal{S}$ relative to the cubical subset $\mathcal{N} \subset \mathcal{X}$, by considering the partial sequences

$$\mathcal{W}^u_{j+1} = \mathcal{F}(\mathcal{W}^u_j) \cap \mathcal{W}^u_j,$$

and

$$\mathcal{W}^s_{j+1} = \mathcal{F}^{-1}(\mathcal{W}^s_j) \cap \mathcal{W}^s_j.$$

For further discussion of the stable and unstable set algorithms, see [DH96], [DH97b], or [DH97a].

The constructive nature of the Theorem 2.4 give rise to practical set-oriented algorithms. We give, for example the pseudo-code for the invariant part algorithm.

ALGORITHM 2.5 (Compute combinatorial invariant part of $\mathcal{N}$).

```
function InvariantPart($\mathcal{N}, \mathcal{F}$)
$\mathcal{S} := \mathcal{N}$;
do
    $\mathcal{A} := \mathcal{S}$;
    $\mathcal{S} := \mathcal{S} \cap \mathcal{F}(\mathcal{S}) \cap \mathcal{F}^{-1}(\mathcal{S})$;
while ($\mathcal{S} \neq \mathcal{A}$)
return $\mathcal{S}$
```

The pseudo-code for

$$\mathcal{W}^s(\mathcal{N}) = \mathbf{localStableSet}(\mathcal{N}, \mathcal{F}),$$

and

$$\mathcal{W}^u(\mathcal{N}) = \mathbf{localUnstableSet}(\mathcal{N}, \mathcal{F}),$$

is similar.

2.3. *A-Posteriori* Validation and Discrete Conley Index Theory. In order to verify that the underlying dynamical system inherits a combinatorial property of its set-oriented discretization, we use the some tools from the discrete Conley Index theory. For more complete discussion of the Discrete Conley Index and its application to rigorous verification see [KMM04, Mis99, KMV05, DJM04a, DJM04b, DFT08].

Consider a continuous map $f : X \subset \mathbb{R}^n \to \mathbb{R}^n$ with X compact.

DEFINITION 2.6. *If $N \subset X$ and $Inv(N, f) \subset N^\circ$, then we say that N is an isolating neighborhood. An invariant set $S \subset X$ is isolated under f if there exists and isolating neighborhood N such that $S = Inv(N, f)$.*

Note that if $S \subset X$ is invariant under f, and $\mathcal{N} \subset \mathcal{X}$ with $S \subset \mathbf{invariantPart}(\mathcal{F}, \mathcal{N})$, then $\mathcal{N}$ isolates S by the outer enclosure property of $\mathcal{F}$.

DEFINITION 2.7. *A topological pair (P_1, P_0) with $P_0 \subset P_1 \subset X$ is called an index pair for f if it satisfies*

- *$Inv(cl(P_1 \backslash P_0, f) \subset int(cl(P_1 \backslash P_0))$ (isolation)*
- *$f(P_0) \cap P_1 \subset P_0$ (positive invariance)*
- *$\partial_f(P_1) \subset P_0$ (exit set)*

Here the *f-boundary* of a set $A \subset X$ is defined to be $\partial_f(A) \equiv cl(f(A) \backslash A) \cap A$.

DEFINITION 2.8. *$\mathcal{N}$ is called a cubical isolating neighborhood of $\mathcal{S}$ in $\mathcal{X}$ if*

$$\mathbf{wrap}(\mathbf{InvariantPart}(\mathcal{N}, \mathcal{F})) \subset \mathcal{N}.$$

If $\mathcal{S}$ is a maximal combinatorial invariant set for which there exist a cubical isolating neighborhood, then $\mathcal{S}$ is said to be an isolated invariant set.

We note that weaker notions of isolation appear in the literature [PS08].

Let $\mathcal{N} \subset \mathcal{X}$ with $\mathcal{X}$ a cubical cover of X, and $\mathcal{F}$ be a combinatorial outer enclosure of f on $\mathcal{X}$. The following combinatorial index pair algorithm, from [KMM04] chapter 10.6, always stops and produces either a "*failure*" or a combinatorial index pair for f.

ALGORITHM 2.9 (Compute Combinatorial Index Pair).

```
function indexPair($\mathcal{N}, \mathcal{F}$)
$\mathcal{S} = \mathbf{invariantPart}(\mathcal{N}, \mathcal{F})$;
$\mathcal{M} = \mathbf{wrap}_{\mathcal{G}}(S)$;
if ($\mathcal{M} \subset \mathcal{N}$);
    $\mathcal{F} = \mathbf{restrict}(\mathcal{F}, \mathcal{M})$;
    $\mathcal{C} = \mathbf{collar}(\mathcal{S})$;
    $\mathcal{P}_0 = \mathcal{F}(\mathcal{S}) \cap \mathcal{C}$;
    do;
        $last\mathcal{P}_0 = \mathcal{P}_0$;
        $\mathcal{P}_0 = \mathcal{F}(\mathcal{P}_0) \cap \mathcal{C}$;
        $\mathcal{P}_0 = \mathcal{P}_0 \cup last\mathcal{P}_0$;
        while ($\mathcal{P}_0 = last\mathcal{P}_0$);
        $\mathcal{P}_1 = \mathcal{S} \cup \mathcal{P}_0$;
        $\bar{\mathcal{P}}_1 = \mathcal{F}(\mathcal{P}_1)$;
        $\bar{\mathcal{P}}_0 = \bar{\mathcal{P}}_1 \backslash \mathcal{S}$;
        return ($\mathcal{P}_1, \mathcal{P}_0, \bar{\mathcal{P}}_1, \bar{\mathcal{P}}_0$) ;
else
        return "failure" ;
endif
```

Let $\mathcal{S} = \textbf{InvariantPart}(\mathcal{N}, \mathcal{F})$. The following theorem is from [KMM04] section 10.6.

Theorem 2.10. *If Algorithm (2.9) returns without failure,*

- *$\mathcal{S}$ is an isolating neighborhood for f.*
- *The pair $(\mathcal{P}_1, \mathcal{P}_0)$ is an index pair for f and isolates $Inv(\mathcal{S}, f)$.*
- *$\mathcal{P}_1 \subset \bar{\mathcal{P}}_1$ and $\mathcal{P}_0 \subset \bar{\mathcal{P}}_0$.*
- *$f(\mathcal{P}_1) \subset \bar{\mathcal{P}}_1$ and $f(\mathcal{P}_0) \subset \bar{\mathcal{P}}_0$.*

The sets $\bar{\mathcal{P}}_0$ and $\bar{\mathcal{P}}_1$ are used in the definition of the index map (see definition 2.12 below).

Note that while the inputs to algorithm 2.9 involve only the combinatorial data $(\mathcal{X}, \mathcal{F})$, and $\mathcal{N}$, the conclusions of Theorem 2.10 tell us about the dynamics of the underlying map f on the topological pair $(\mathcal{P}_1, \mathcal{P}_0)$. This is a first step toward obtaining rigorous information about f from its combinatorial enclosure. However, Algorithm 2.9 and Theorem 2.10 are insufficient, as the theorem does not guarantee that $\mathcal{S} \neq \emptyset$.

The following lemma can be found in [KMM04] section 10.6;

Lemma 2.11. *if f, $\mathcal{P}_1, \mathcal{P}_0$, $\bar{\mathcal{P}}_1$, and $\bar{\mathcal{P}}_0$ are as in Algorithm 2.9. Then the homomorphism induced on relative homology by the inclusion map $\iota : (\mathcal{P}_1, \mathcal{P}_0) \hookrightarrow (\bar{\mathcal{P}}_1, \bar{\mathcal{P}}_0)$ is in fact an isomorphism. It follows that*

$$\iota_*^{-1} : H_*(\bar{\mathcal{P}}_1, \bar{\mathcal{P}}_0) \to H_*(\mathcal{P}_1, \mathcal{P}_0)$$

is well defined.

By Theorem 2.10, f is a pair map and induces a homomorphism on relative homology, which we denote by $f_* : H_*(\mathcal{P}_1, \mathcal{P}_0) \to H_*(\bar{\mathcal{P}}_1, \bar{\mathcal{P}}_0)$. From a computational point of view, it is essential that the induced homomorphism f_* can be computed only from the combinatorial data $\mathcal{F}$. See Theorem 7.15 in [KMM04] for technical details, and [Pil97] for numerical implementation.

Definition 2.12. *Define the index map $f_{\mathcal{P}_*}$ of f relative to the pair $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_0)$ by*

$$f_{\mathcal{P}_*} = \iota_*^{-1} \circ f_* : H_*(\mathcal{P}_1, \mathcal{P}_0) \to H_*(\mathcal{P}_1, \mathcal{P}_0)$$

The index map is the basis of the definition of the Discrete Conley Index. See for example [Mis99] or [KMM04]. In the present work however, we will not need the full power of the Conley Index. Rather we exploit the following notion of *Lefschetz number* on pairs.

Definition 2.13. *If $S \subset \mathbb{R}^n$ be an isolated invariant set for the map f, and $\mathcal{P}$ be any index pair for S. The Lefschetz number of S relative to $\mathcal{P}$ is defined to be*

$$L(S, f) = \sum_{j=0}^{n} (-1)^j tr(f_{\mathcal{P}_*}).$$

Theorem 2.14. *If $L(S,f) \neq 0$, one has not only that $inv(S,f) \neq \emptyset$ but that the invariant set actually contains a fixed point.* See [Srz00] for the proof and details.

We close this section with an example application of the discrete Conley Index machinery to the problem of 'chaos verification'.

Example 2.15 (Verification of a Topological Horseshoe Factor.).

Definition 2.16. *The dynamical system (M,f) is said to admit a topological horseshoe factor if there exists a compact subset $X \subset M$ so that $f|_X$ semi-conjugate to Σ_A, where Σ_A is a sub-shift of finite type with positive topological entropy.*
For the formal definition of sub-shift of finite type, and topological entropy see [Kat] chapters 1.9 and 3.1 respectively.

Let $\mathcal{S}$, and $(\mathcal{P}_1,\mathcal{P}_0)$ be as in Algorithm 2.9.

Theorem 2.17. *Suppose that $(\mathcal{P}_1,\mathcal{P}_0) \subset \mathcal{X}$ is the finite union of disjoint, compact pairs, $(\mathcal{P}_1,\mathcal{P}_0)_1$, …, $(\mathcal{P}_1,\mathcal{P}_0)_m$, and define the sets $N_i = cl(\mathcal{P}_1 \backslash \mathcal{P}_0)$. Denote by f_{N_i} the map $f|_{N_i}$, and by $\{i_1,\ldots,i_K = i_1\}$ a finite length-K periodic sequence of the numbers $1,\ldots,m$. Then if*

$$L(N_j, f_{N_{i_1}} \circ \ldots \circ f_{N_{i_K}}) \neq 0$$

there exists a periodic orbit of period K in $\mathcal{S}$ that passes through the regions N_{i_1}, …, N_{i_K} in $i_1,\ldots,i_K$ order.

The result can be found in [DFT08], and [Srz00], along with the proof. Define the preliminary $m \times m$ transition matrix $\bar{A}$ for the index pair by

$$\bar{a}_{ij} = \begin{cases} 1 & \text{if} \quad \mathcal{F}(N_j) \cap N_i \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

These are the combinatorial, or potential connecting orbits. Theorem 2.17 is used to verify the existence of f orbits corresponding to the transitions suggested by $\bar{A}$. Since it may be impossible to verify all the potential transitions, one typically has to settle for a smaller matrix A of verified transitions where A contains the entries of $\bar{A}$ which correspond cycles having non-zero Lefschetz index. The topological entropy of $\sigma_A : \Sigma_A \to \Sigma_A$ is equal to the natural logarithm of the spectral radius of A (see [Kat] 3.2). If the topological entropy of σ_A is greater than zero, then f has a horseshoe factor.

A detailed example of the argument just sketched is found in [KMM04] chapter 10.6. In [DFT08], the authors develop algorithms which automate this analysis. Using their automated procedure they are able to verify a semiconjugacy between an index pair for the the Henon map and a subshift of finite type on 199 symbols.

3. Top-Down Set-Oriented Computations for Conservative Dynamical Systems.

3.1. Top-Down Computation of Combinatorial Connecting Orbits. In this section we present Algorithm 3.7, the main result of the paper. The algorithm executes a top-down search for combinatorial connecting orbits between hyperbolic objects. This constitutes an implementation of the heuristic set-oriented search mentioned in Algorithm 1.3 Here follows an outline.

Algorithm 3.1 (Connecting orbit meta-algorithm).

Step 1: Local computation about some hyperbolic object.
Step 2: Globalize stable and unstable sets and cover some connecting orbits.
Step 3: Restrict, subdivide, and repeat.

In order to formalize this procedure, we focus on computing homoclinic orbits for a hyperbolic fixed point and elaborate on each step. Some modifications are discussed in Section 4.

Step 1: (Local Computation) Let p be a hyperbolic fixed point of a continuous map $f : X \subset \mathbb{R}^n \to \mathbb{R}^n$. Let $\mathcal{X}_0$ be a uniform cubical cover of X, and $\mathcal{F}_0 = \mathbf{enclose}(\mathcal{X}_0, f)$. Take $\mathcal{P}_0$ to be the smallest collection of cubes covering p.

We must assume that $\mathcal{X}_0$ is of high enough resolution that the following isolation condition is met: We require that there is a connected cubical set $\mathcal{FP}_0$ with $\mathcal{P}_0 \subset \mathcal{FP}_0$ and such that $\mathcal{FP}_0 = \mathbf{invariantPart}(\mathbf{wrap}(\mathcal{FP}_0), \mathcal{F}_0)$. In other words, $\mathcal{FP}_0$ is the smallest cubical isolated invariant set containing p.

Example 3.2. *The following example illustrates the need for the distinction between $\mathcal{P}_0$ and $\mathcal{FP}_0$. Consider the linear map given by the matrix*

$$L = \begin{pmatrix} 10/9 & 0 \\ 0 & -9/10 \end{pmatrix},$$

and take $\mathcal{X}_0$ to be a cubical complex covering $[-2,2]^2$ with resolution $r = 0.5$. Note that $\mathcal{P}_0$ is the four cubes making up the square $[-0.5, 0.5] \times [-0.5, 0.5]$.

To see that $\mathcal{P}_0$ is not isolated, let $Q = [0, 0.5] \times [0.5, 1] \in \mathcal{X}_0$ and note that $Q \notin \mathcal{P}_0$. Q contains the point $x = (0, 0.8)$, and since $L^2(x) = (0, 0.64) \in Q$, the cube Q has combinatorial period two. While this period two trajectory is a false positive, Q is nevertheless combinatorially invariant. Since $Q \in \mathbf{wrap}(\mathcal{P}_0)$, the combinatorial fixed point set in this example is not isolated at this level of resolution.

Since we do not want to consider $\mathcal{FP}_0$ as part of our initial data, we give an algorithm for finding it.

Algorithm 3.3 (Compute isolated invariant component of combinatorial fixed cubes).

```
function invariantComponent ($\mathcal{P}, \mathcal{X}, \mathcal{F}$)
$\mathcal{FP} := \mathcal{P}$;
do
    $\mathcal{A} := \mathcal{FP}$;
    $\mathcal{B} := \mathbf{wrap}(\mathcal{A})$;
    $\mathcal{C} := \mathbf{invariantPart}(\mathcal{B}, \mathcal{F})$;
    $\mathcal{FP} := \mathcal{C}$;
while ($\mathcal{A} \neq \mathcal{FP}$)
if ($\mathcal{FP} \cap \partial\mathcal{X} \neq \emptyset$);
    return "failure";
else
    return $\mathcal{FP}$
end
```

The algorithm stops as $\mathcal{X}_0$ is a finite collection of cubes, and fails if $\mathcal{P}_0$ cannot be isolated in $\mathcal{X}_0$.

Now that we have isolated the fixed point, the next step in the local computation is to compute a larger cubical neighborhood which still isolates $\mathcal{FP}_0$.

Algorithm 3.4 (Expand the isolating neighborhood about $\mathcal{FP}$).

```
*function* growIsolatingBlock ($\mathcal{FP}, \mathcal{X}, \mathcal{F}$)
$\mathcal{S} := \mathcal{FP}$;
$\mathcal{W} := \mathbf{wrap}(\mathcal{S})$;
$\mathcal{S}' := \mathbf{invariantPart}(\mathcal{W})$;
*while* ($\mathcal{S} = \mathcal{S}'$);
    $\mathcal{S} := \mathcal{W}$;
    $\mathcal{W} := \mathbf{wrap}(\mathcal{S})$;
    $\mathcal{S}' := \mathbf{invariantPart}(\mathcal{W})$;
*end while*
$\mathcal{N} := \mathcal{S}$;
*if* ($\mathcal{N} = \mathcal{FP}$);
    *return* "failure";
*else*
    *return* $\mathcal{N}$
*end*
```

This algorithm wraps $\mathcal{FP}_0$ until a further wrap would introduce new invariant cubes. If $\mathcal{FP}_0$ cannot be isolated then the algorithm fails. However if $\mathcal{FP}_0$ is the output of algorithm 3.3, then $\mathcal{FP}_0$ is preconditioned for successful execution of algorithm 3.4.

Remark 3.5 (Global Meaning of $\mathcal{N}_0$). *Consider the dynamics on $\mathcal{N}_0$. Apply the stable and unstable set algorithms to $\mathcal{N}_0$ and obtain both*

$$\mathcal{W}_0^u \equiv \mathbf{localUnstableSet}(\mathcal{N}_0, F_0),$$

and

$$\mathcal{W}_0^s \equiv \mathbf{localStableSet}(\mathcal{N}_0, F_0).$$

If

$$x \in Q \in \mathcal{N}_0 \backslash \left(\mathcal{W}_0^s \cup \mathcal{W}_0^u\right), \tag{3.1}$$

then the pointwise orbit of x under the true dynamics f leaves $\mathcal{N}_0$ in a finite number of both forward and backward iterations. By Remark c in Section 2.1, the subdivision operation can increase neither an invariant set, nor its local stable or unstable sets. Then Equation 3.1 persists as we increase the resolution, and any homoclinic orbits of p pass through $\mathcal{W}_0^s \cup \mathcal{W}_0^u$. It follows that the cubes in $\mathcal{N}_0 \backslash \left(\mathcal{W}_0^s \cup \mathcal{W}_0^u\right)$ are of no further interest throughout the remainder of the computation.

Step 2: (Globalize) The idea now is to iterate the stable and unstable cubical sets until we are confident that their intersection covers a heteroclinic connection. An efficient globalization algorithm must strike a balance between two considerations; globalize long enough to insure that a connection is covered, but not so long that the intersection contains too many unwanted cubes.

Our globalization scheme is guided by the λ-lemma, which says that the stable and unstable manifolds must accumulate on themselves whenever there is a tangle. This provides the following heuristic stopping condition: we globalize the stable and unstable sets until the globalization returns to the local block $\mathcal{N}_0$. Once this occurs, we are confident that a homoclinic excursion is covered. (Of course, a full verification will be obtained in the second stage of the computation).

This choice is formalized in the following algorithm.

ALGORITHM 3.6 (globalization algorithm for the local unstable cover).

```
function globalizeUnstableSet ($\mathcal{X}, \mathcal{W}^u_{loc}, \mathcal{N}_{loc}, \mathcal{F}$)
$\mathcal{A} := \mathbf{wrap}_{\mathcal{X}}(\mathcal{W}^u_{loc}) \cap \mathcal{N}_{loc}$;
$\mathcal{B} := \mathcal{A} \backslash \mathcal{W}^u_{loc}$;
$\mathcal{C}^u := \mathcal{F}(\mathcal{W}^u_{loc})$;
while ($\mathcal{C}^u \cap \mathcal{B} = \emptyset$);
        $\mathcal{C}_{last} = \mathcal{C}^u$;
        $\mathcal{C}^u := \mathcal{F}(\mathcal{C}^u)$;
        if ($\mathcal{C}^u = \mathcal{C}_{last}$)
                return "failure";
        end if
end while
return $\mathcal{C}^u$
```

This algorithm stops as $\mathcal{F}^k(\mathcal{W}^u_{loc})$ is eventually constant ($\mathcal{X}_0$ consists of only finitely many cubes). The algorithm fails if the sequence becomes constant without intersecting $\mathcal{B}$. The globalization of the stable set is simply

$$\mathcal{C}^s_0 = \mathbf{globalizeUnstableSet}(\mathcal{W}^s_0, \mathcal{N}_0, \mathcal{F}^{-1}).$$

Now let

$$\mathcal{I}_0 = (\mathcal{C}^u_0 \cap \mathcal{C}^s_0) \backslash \mathcal{N}_0,$$

and

$$\mathcal{D}_0 = \mathcal{W}^u_0 \cup \mathcal{W}^s_0.$$

$\mathcal{I}_0 \cup \mathcal{D}_0$ is our candidate for a cubical cover of a connecting orbit for p.

Step 3: (Restrict, Subdivide and Repeat) Let

$$\mathcal{X}_1 = \mathbf{subdivide}(\mathcal{I}_0 \cup \mathcal{D}_0),$$

define the new domain for the next loop through Algorithm 3.1. To loop the algorithm we repeat the steps from the base case (just described), with one important difference: there is no need to recompute the isolating neighborhood, as any subdivision of $\mathcal{N}_0$ continues to isolate p. We let

$$\mathcal{N}_1 = \mathbf{subdivide}(\mathcal{D}_0).$$

and throw away all data from the earlier stage of the computation. Iteration of the algorithm proceeds exactly as just described for all further subdivisions.

This discussion is formalized in following pseudo-code. The input to the algorithm is the cubical cover $\mathcal{X}_0$, the fixed cubes $\mathcal{P}_0$, the map f, and the stopping resolution ϵ_f.

ALGORITHM 3.7 (Top down search for a combinatorial homoclinic excursion).

```
function connectingOrbitSearch ($\mathcal{X}_0, \mathcal{P}_0, \epsilon_f, f$)
$\mathcal{F} := \mathbf{enclose}(\mathcal{X}_0, f)$;
$\mathcal{FP} := \mathbf{invariantNeighbors}(\mathcal{P}_0, \mathcal{F})$;
```

```
$\mathcal{N}$ := **growIsolatingBlock**$(\mathcal{FP}, \mathcal{F})$;
$\mathcal{W}^s_{loc}$ := **localStableSet**$(\mathcal{N}, \mathcal{F})$;
$\mathcal{W}^u_{loc}$ := **localUnstableSet**$(\mathcal{N}, \mathcal{F})$;
$\mathcal{C}^s$ := **globalizeStableSet**$(\mathcal{W}^s_{loc}, \mathcal{N}_{loc}, \mathcal{F})$;
$\mathcal{C}^u$ := **globalizeStableSet**$(\mathcal{W}^u_{loc}, \mathcal{N}, \mathcal{F})$;
$\mathcal{D} := (\mathcal{W}^s_{loc} \cup \mathcal{W}^u_{loc})$;
$\mathcal{I} := (\mathcal{C}^s \cap \mathcal{C}^u) \backslash \mathcal{D}$;
*while(*diameter$(\mathcal{I}) \geq \epsilon_f)$;
        $\mathcal{X}$ := **subdivide**$(\mathcal{I} \cup \mathcal{D})$;
        $\mathcal{N}$ := **subdivide**$(\mathcal{D})$;
        $\mathcal{F}$ := **enclose**$(\mathcal{X}, f)$;
        $\mathcal{W}^s_{loc}$ := **localStableSet**$(\mathcal{N}, \mathcal{F})$;
        $\mathcal{W}^u_{loc}$ := **localUnstableSet**$(\mathcal{N}, \mathcal{F})$;
        $\mathcal{C}^s$ := **globalizeStableSet**$(\mathcal{W}^s_{loc}, \mathcal{N}, \mathcal{F})$;
        $\mathcal{C}^u$ := **globalizeStableSet**$(\mathcal{W}^u_{loc}, \mathcal{N}, \mathcal{F})$;
        $\mathcal{D} := (\mathcal{W}^s_{loc} \cup \mathcal{W}^u_{loc})$;
        $\mathcal{I} := (\mathcal{C}^s \cap \mathcal{C}^u) \backslash \mathcal{D}$;
*end while*
*return* connectingOrbit := **invariantPart**$(\mathcal{I} \cup \mathcal{D})$;
*end;*
```

The algorithm fails if any of its sub-functions fail. The algorithm always stops as $\epsilon_f > 0$ and the subdivide operator reduces the resolution at a geometric rate.

REMARKS 3.8.

1. *In practice, $\mathcal{P}_0$ can be computed from the dynamical graph by considering nodes with self connections.*
2. *If $\mathcal{X}_0$ contains a large number of cubes, the process of repeatedly wrapping a set and computing the invariant part of the wrap is computationally expensive. However Algorithm 3.4 is called only once during the initial stage of Algorithm 3.7.*
3. *If* **growIsolatingNbhd** *returns $\mathcal{X}_0$, then the only invariant cubes in $\mathcal{X}_0$ are $\mathcal{FP}_0$, and there is no possibility of detecting a horseshoe. This means either that $\mathcal{X}_0$ does not cover a horseshoe, or that all the interesting dynamics are covered by $\mathcal{FP}_0$ (in which case the initial resolution is too large).*
4. *Algorithms 3.3 and 3.4 determine the needed balance between the initial number of cubes, and the initial resolution. The grid must be fine enough that isolation is achieved in algorithm 3.3, but not so fine that algorithm 3.4 is impractical. This trade off is used in order to chose a satisfactory initial grid.*
5. *The geometric stopping condition in the globalization step, is decidedly different from the choice of 'first intersection' employed in [DJT01]. In [DJT01], the unreliability of the 'first intersection' is exploited to define a hat-function, which picks out parameter values at which bifurcations occur. In our case however this unreliability of first intersections will destroy our subdivision scheme. The purpose of the geometric globalization is to increase our confidence that a connecting orbit is covered, before restriction and subdivision.*
6. *Since we are assuming that $\mathcal{FP}_0$ covers the hyperbolic fixed point p, it follows that $\mathcal{W}^{u,s}_0$ actually cover the local stable and unstable manifolds of p (see [DH97b], and [DH97a]).*
7. *Experimentation is often needed in order choose an effective stopping condition ϵ_f. For the examples in Section 4 we subdivide until $\mathcal{I}_m$ is composed of*

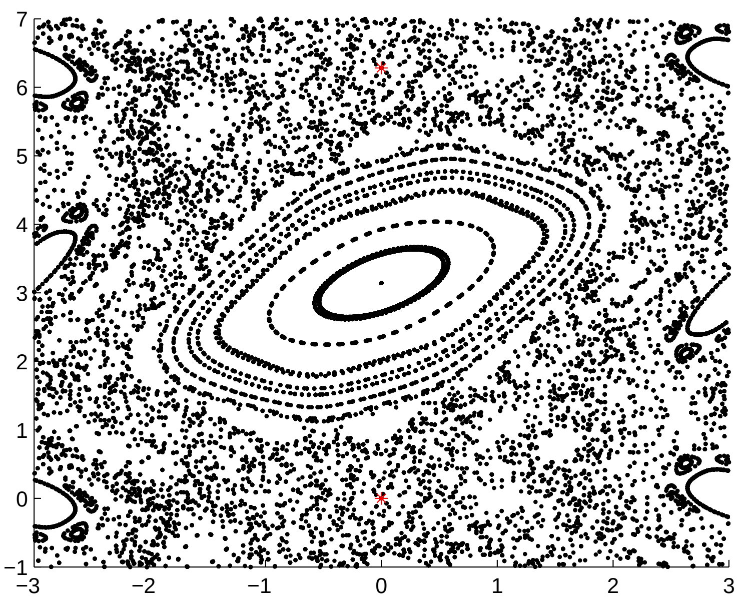

Fig. 4.1. *Phase Space Sample for the Standard Map.*

several distinct topological components. This indicates that invariant part of $\mathcal{X}_m$ covers only the desired zero dimensional set (see Figure 4.6). Vagueness of stopping condition is present in most automated chaos verification schemes, and is not a idiosyncracy of the present work. (See [DJM04a], [DFT08], and [DJM05]).

8. *The functions in Algorithm 3.7 which might fail are* **invariantNeighbors***, and the globalization calls. The failure of the first indicates too high an initial resolution. The failure of the second indicates that $\mathcal{X}_0$ was chosen too small to cover a homoclinic tangle.*

4. Example Computations.

4.1. Example 1: Homoclinic Tangle at the Origin in the Standard Map. Consider the family of maps $f_\epsilon : \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$f_\epsilon(x, y) = \begin{pmatrix} x + \epsilon \sin(y) \\ x + y + \epsilon \sin(y) \end{pmatrix}.$$

f_ϵ is called the standard map, and is the subject of a substantial literature. Fig (4.1) shows a sample of the phase space dynamics for $\epsilon = 1.2$. At this parameter value the phase space is dominated by secondary the KAM tori in the resonance zone about the elliptic fixed point $(0, 2\pi)$, and the Birkhoff instability zone containing the hyperbolic fixed point at $p = (0, 0)$. Note that due to the presence of invariant circles, the dynamics of the map is neither minimal nor mixing. Nevertheless we expect an abundance of chaotic orbits in the Birkhoff zone.

Set $X = [-2, 8] \times [-2, 8]$, and let $\mathcal{X}_0$ be a 50×50 cubical grid covering X. Then this grid has resolution is $r = 0.2$ and contains 2500 cubes. Direct computation confirms

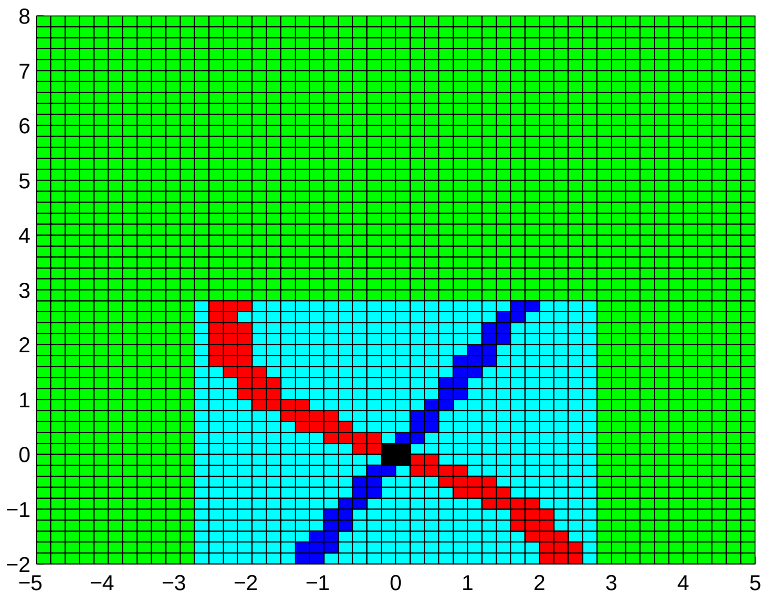

Fig. 4.2. *Compute Local Picture: The combinatorial fixed point $\mathcal{FP}_0$ (black), the local block $\mathcal{N}_0$ (light blue), as well as the local stable and unstable sets $\mathcal{W}^s_{\mathcal{N}_0}(\mathcal{FP}_0)$ and $\mathcal{W}^u_{\mathcal{N}_0}(\mathcal{FP}_0)$ (red and dark blue respectively).*

that the resolution is low enough that **invariantNeighbors** and **growIsolatingNbd** execute successfully, yet high enough that they run in negligible time. We impose a stopping resolution of $\epsilon_f = 0.0065$.

Remarks 4.1 (Set-Oriented Computation Details). *Figures 4.2 - 4.6 illustrate graphically the major steps of mata-algorithm 3.1.*

1. *Figure 4.2 shows the results of the local computation and highlights graphically comment 3.5 from Section 3.1. The point is that any orbit homoclinic to p must pass through the local stable and unstable sets of $\mathcal{N}_0$. Then the remainder of the local block is of no interest.*
2. *Figure 4.4 shows the state of the computation at the end of the first pass through the meta-algorithm. Note that the resonance zone has already been culled from the computation, even though the location of the resonance is not an explicit input to the algorithm. This is an essential feature of our scheme (Compare with Figures A.1 and A.2.)*
3. *Figure 4.5 shows the set I from algorithm (3.7) just before exit from the while loop. Note that the covering has resolved itself into several distinct pieces.*
4. *Figure 4.6 shows the output of Algorithm 3.7 for the standard map example.*

Remarks 4.2 (Empirical Complexity of the Algorithm). *Table 4.1 chronicles the complexity of the search.*

1. *A "brute force" computation with an initial resolution of* $h = 0.00625$ *(the final resolution of the iterative scheme) on the original domain of* $[-5, 5] \times [-2, 7]$ *would required a* 1600×1600 *grid of* 2,560,000 *domain cubes, and as many enclosure computations.*

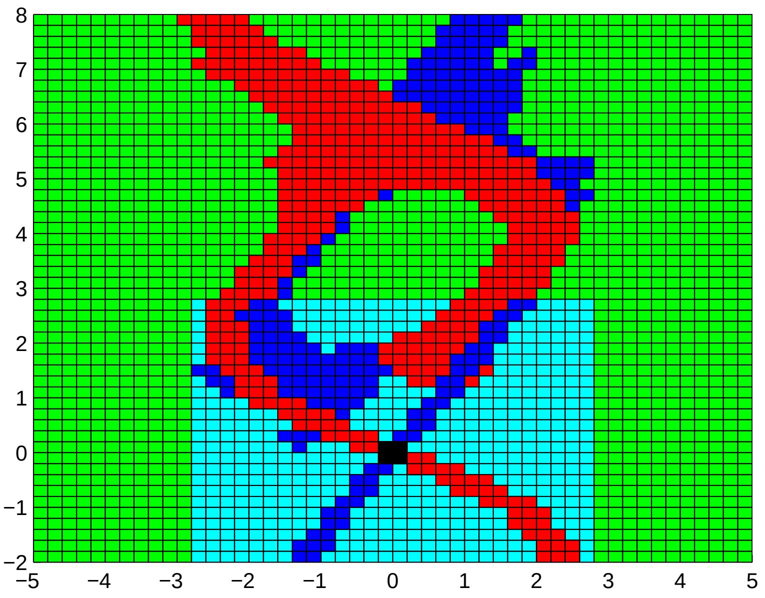

FIG. 4.3. *Globalize: Iterate the local stable and unstable sets until they return to the local block. Note that the global unstable set (in blue) is underneath the global stable set (in red). This is confused slightly by the fact that the local unstable and stable sets (blue and red within the magenta box) are on top of the global unstable and stable sets once they return to the local (magenta) block.*

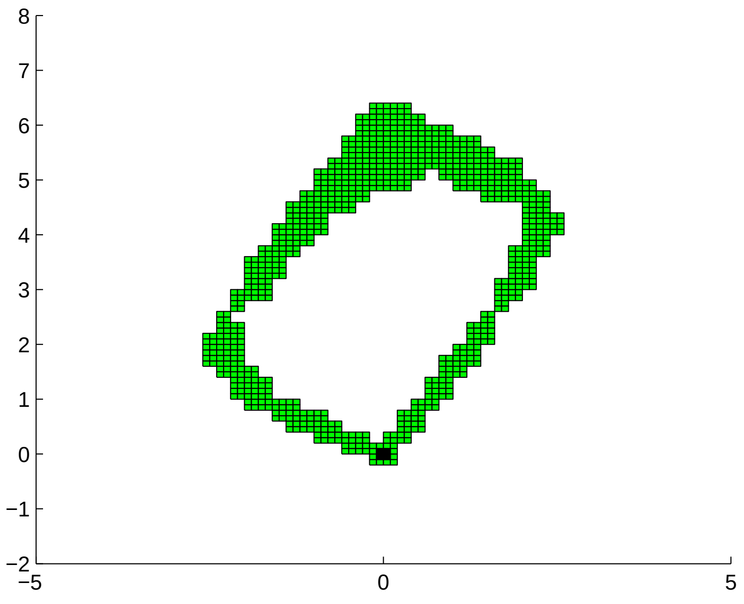

FIG. 4.4. *Restrict and Subdivide: $\mathcal{X}_1$ (with $\mathcal{FP}_1$ also shown).*

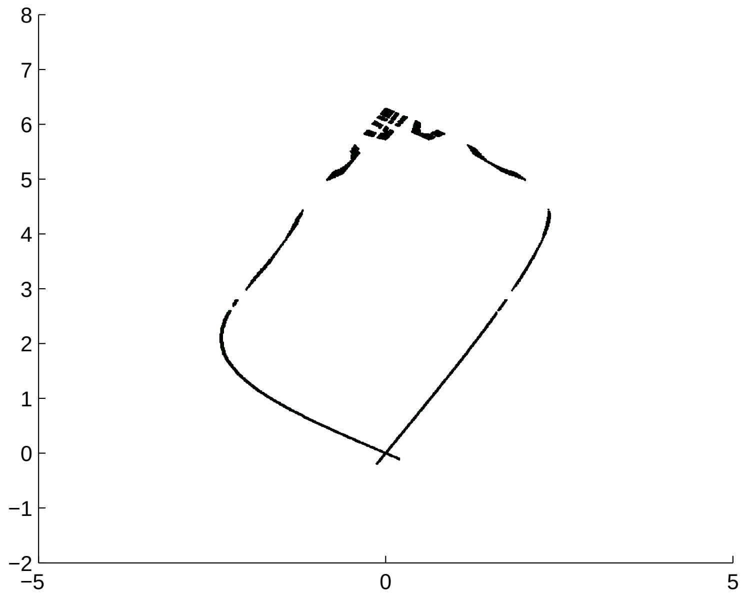

FIG. 4.5. $\mathcal{X}_5$*:In* $\mathcal{X}_5$ *the intersection is resolved into several clear disjoint pieces.*

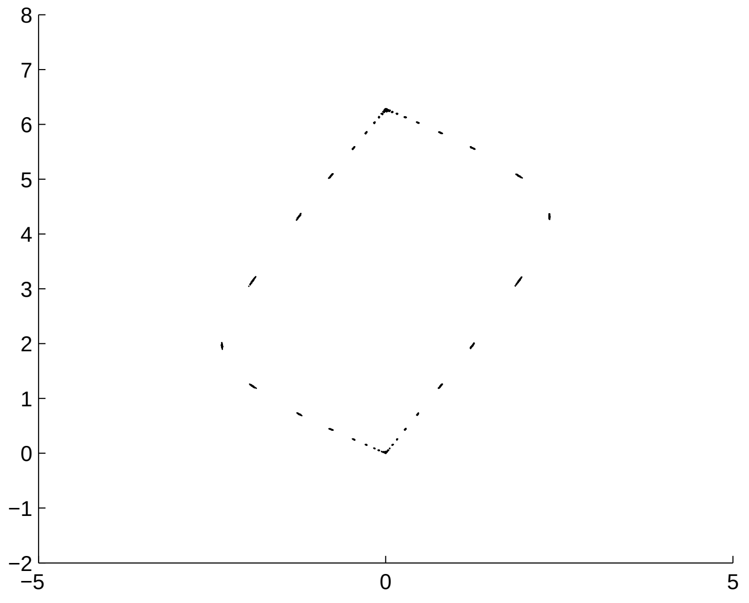

FIG. 4.6. *Algorithm 3.7 Output: Combinatorial approximation to the homoclinic excursion in the standard map.*

2. *With the top-down search, the homoclinic excursion was obtained with only* 30,400 *enclosure computations, or one percent of brute force requirements.*
3. *The largest set arising in the top-down computation contained* 13,400 *cubes. This is less than one percent of the size of the brute force grid. Since our implementation of the union and intersection operations are order N in the set size, this reduction makes a substantial difference in the run time of the program.*
4. *The final cover $\mathcal{S}$ of the homoclinic connection contains roughly* 2,200 *cubes. This is fewer cubes than the initial domain.*

Remarks 4.3 (Post-Processing and Verification). *In order to obtain rigorous results, $\mathcal{S} =$* **invariantPart**$(\mathcal{X}_5)$ *is passed into Algorithm 2.9, which executes successfully. The result is an index pair $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_0)$ for $\mathcal{S}$ (See also the comments in Section 5.3).*

1. *$\mathcal{P}$ and $\mathcal{F}$ are passed into the program 'homcubes' (part of the CHomP suite), which computes the relative homology of the pair, as well as the induced homomorphism.*
2. *The relative homology is*

$$\begin{aligned} H_0(\mathcal{P}_1, \mathcal{P}_0) &= 0 \\ H_1(\mathcal{P}_1, \mathcal{P}_0) &= \mathbb{Z}^{24} \\ H_2(\mathcal{P}_1, \mathcal{P}_0) &= 0. \end{aligned}$$

   Intuitively, the relative homology of the pair is isomorphic to the reduced homology of the topological space obtained by taking the quotient of the index pair by its exit set. The quotient has one topological component, 24 *non-trivial* 1*-generators, no* 2*-generators (a space homotopic to the wedge of twenty four circles).*
3. *We represent the index map by listing its action on the* 24 *generators $H_1(\mathcal{P}_1, \mathcal{P}_0)$. The result is shown in Table 4.1.*
4. *Note that in this example, generator* 14 *is mapped to itself, indicating that the topological component associated generator* 14 *covers the fixed point.*
5. *One can check that the following orbit on generators is present in the index map:*

$$14 \to 12 \to 21 \to -19 \to -6 \to 24 \to -18 \to -4$$

$$\to 22 \to 15 \to 16 \to 23 \to 20 \to 5 \to -3 \to 10 \to 17 \to 1 \to 14.$$

   We refer to such an algebraic orbit as a generator excursion.
6. *The generator excursion indicates the existence of a homoclinic excursion in the underlying dynamics. To complete the argument and rigorously establish that the index pair isolates a topological horseshoe factor, one follows the sketch described in section* (2.3)*.*

The intuitive interpretation of the algebraic data is that A) each topological component of $\mathcal{P}$ gives rise to roughly one generator in the first level of the relative homology, and B) f_{P_*} describes how these generators, and hence the underlying topological components, are mapped across each other under f. A) is because each topological component of the index pair corresponds roughly to an intersection of the stable and

Subdivisions	number of Cubes	resolution
0	2,500	0.2
1	828	0.1
2	1,860	0.05
3	4,092	0.025
4	7,720	0.0125
5	13,400	0.00625

TABLE 4.1
Resolution and Cube Count

$f_{P_*}(\alpha_1)$	$=$	$\alpha_{12}+\alpha_{14}$	$f_{P_*}(\alpha_{13})$	$=$	α_3
$f_{P_*}(\alpha_2)$	$=$	α_{21}	$f_{P_*}(\alpha_{14})$	$=$	$\alpha_{12}+\alpha_{14}$
$f_{P_*}(\alpha_3)$	$=$	α_{10}	$f_{P_*}(\alpha_{15})$	$=$	α_{16}
$f_{P_*}(\alpha_4)$	$=$	α_{22}	$f_{P_*}(\alpha_{16})$	$=$	α_{23}
$f_{P_*}(\alpha_5)$	$=$	$-\alpha_3$	$f_{P_*}(\alpha_{17})$	$=$	$\alpha_1+\alpha_2$
$f_{P_*}(\alpha_6)$	$=$	α_{24}	$f_{P_*}(\alpha_{18})$	$=$	$-\alpha_4$
$f_{P_*}(\alpha_7)$	$=$	α_{10}	$f_{P_*}(\alpha_{19})$	$=$	$-\alpha_6$
$f_{P_*}(\alpha_8)$	$=$	0	$f_{P_*}(\alpha_{20})$	$=$	$\alpha_5-\alpha_8$
$f_{P_*}(\alpha_9)$	$=$	α_{10}	$f_{P_*}(\alpha_{21})$	$=$	$-\alpha_{19}$
$f_{P_*}(\alpha_{10})$	$=$	α_{17}	$f_{P_*}(\alpha_{22})$	$=$	α_{15}
$f_{P_*}(\alpha_{11})$	$=$	0	$f_{P_*}(\alpha_{23})$	$=$	$\alpha_{11}+\alpha_{20}$
$f_{P_*}(\alpha_{12})$	$=$	α_{21}	$f_{P_*}(\alpha_{24})$	$=$	$-\alpha_{18}$

TABLE 4.2
Induced homomorphism in the standard map; The action of f_{P_*} on the generators of $H_*(\mathcal{P}_1,\mathcal{P}_0)$

unstable manifolds. The exit sets are due to the stretching of components in the unstable direction. The collapse of the exit set of a component produces a generator on the first level of homology. Then B) suggests that f_{P_*} maps the generator α_i associated with a particular topological component $\mathcal{N}_i$ of $\mathcal{P}$, to the generators $f_{P_*}(\alpha_i)$ of the topological components that are accessible from $\mathcal{N}_i$ under the application of f.

4.2. Heteroclinic Tangle in the The Suris Map.

The Suris map is defined by

$$f(\theta,r)=\begin{pmatrix}\theta+r+V'(\theta)+\epsilon P(\theta)\\ r+V'(\theta)+\epsilon P(\theta)\end{pmatrix}$$

where

$$V'(\theta)=-\frac{2}{\pi}atan\left(\frac{\delta\sin(2\pi\theta)}{1+\delta\cos(2\pi\theta)}\right)$$

and

$$P(\theta)=2\pi\cos(\pi\theta)\sin(\pi\theta)$$

This map was presented in [Sur94] as an example a family of area preserving twist maps which is integrable when $\epsilon=0$, and where the integrable map contains both elliptic and hyperbolic fixed points. In the integrable system $W^s(p_1)=W^u(p_2)$ and $W^u(p_1)=W^s(p_2)$ so the phase space of the integrable map is qualitatively similar to

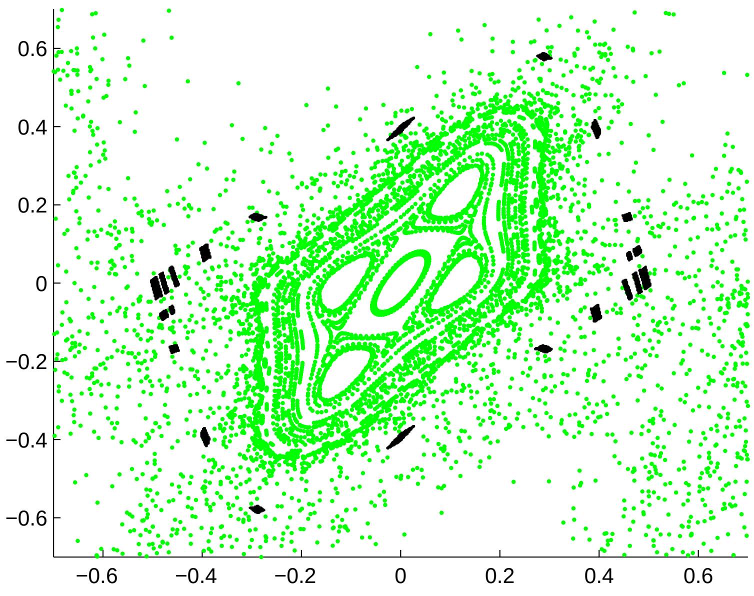

FIG. 4.7. *A phase space sample for the Suris map, as well as an index pair* $(\mathcal{P}_1, \mathcal{P}_0)$ *for a topological horseshoe factor between two hyperbolic fixed points.*

the phase space of the mathematical pendulum. Numerics suggest that for $\epsilon > 0$ the manifolds split, and a heteroclinic tangle develops.

Consider the map at the parameter values $\delta = 0.4$ and $\epsilon = 0.05$. The notable difference between the present computation and algorithm 3.7 is that here we are trying to compute a heteroclinic excursion. The present computation follows the same meta-search outline given in 3.1, with the difference being that the local computations are performed about two distinct fixed points. Then, in the globalization step the stable and unstable sets of one fixed point are continued into the neighborhood of the other, rather than back into themselves.

The computation proceeds as in Section 4.1. For this computation we took an initial region $X = [-1.5, 1.5]^2$, an initial resolution of $\epsilon_0 = 0.03$, and require a final resolution less than $\epsilon_f = 0.002$. The algorithm converges after four iterations to a combinatorial connection $\mathcal{S}$ between p_1 and p_2. The resulting index pair is depicted in Fig. 4.7.

4.3. Heteroclinic Tangle in the Area Preserving Henon Map. As a final planar example, consider the family of quadratic mappings

$$f(x, y) = \begin{pmatrix} 1 + y - ax^2 \\ x \end{pmatrix}$$

This is the area preserving Henon Family. We take $a = 1.2$, putting the family close to the "anti-integrable" limit. At this parameter value no elliptic behavior is readily observable, and naive phase space sampling of trajectories yields little useful information about the dynamics. Nevertheless the map admits a pair of hyperbolic

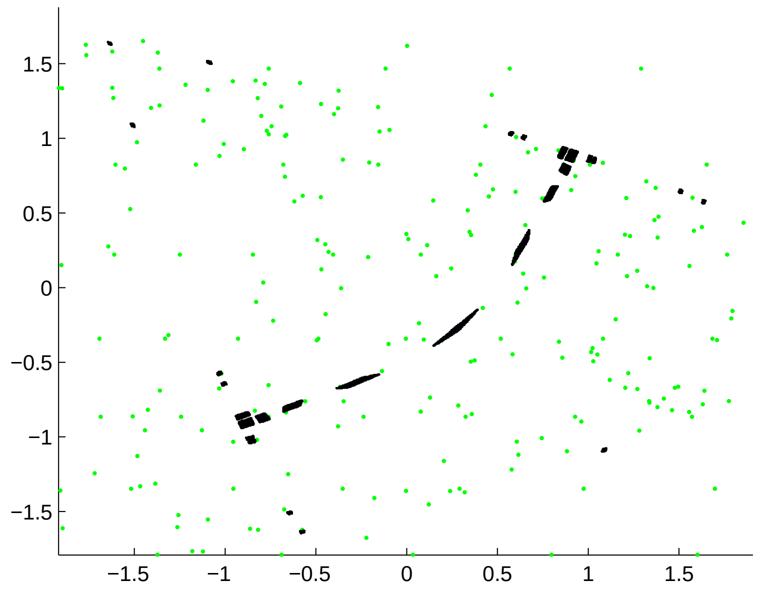

Fig. 4.8. *A phase space sample for the area preserving Henon map, as well as an index pair $(\mathcal{P}_1, \mathcal{P}_0)$ about the hyperbolic fixed points. While the phase space sample does not reveal any regular dynamics, the search is able to localize the horseshoe.*

fixed points; $p_1 = (-1/\sqrt{a}, -1/\sqrt{a})$, and $p_2 = (1/\sqrt{a}, 1/\sqrt{a})$. If the stable and unstable manifolds of these intersect transversally, there is still chaos in the region.

Even though we know exactly where the hyperbolic points are located, we do not use this information in the program. Instead we begin with a 50×50 grid on the square $X = [-7, 7]^2$ and have the program find the fixed cubes as well as the connecting trajectories along the lines discussed in Section 3.1 (the point is that the algorithm is given only very rough and qualitative information about where to look for the connecting orbits).

We take an initial resolution of $\epsilon_0 = 0.28$ and require a final resolution of less than $\epsilon_f = 0.005$. The top-down search stops after six subdivisions and returns a combinatorial connection $\mathcal{S}$, which we use to grow the index pair $(\mathcal{P}_1, \mathcal{P}_0)$. The index pair, as well as a number of pointwise trajectories are shown in Fig 4.8.

Remark 4.4. *Observe that the symbolic dynamics in this example are somewhat more complicated than in the previous examples. As in the case of the Suris map there is one connection from p_1 to p_2. However, for this Henon map there are two connections from p_2 back to p_1, and the system is semi-conjugate to a sub-shift on three symbols. This added complexity is handled automatically by the top-down search.*

4.4. Homoclinic Tangle in the Volume Preserving ABC Map. Although the algorithms run faster, and the pictures are easier to render and view in the plane, the utility of the scheme developed in this work is not limited to the study of two dimensional maps. Consider the three parameter family of maps defined by

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{pmatrix} = \begin{pmatrix} x_n + A\sin(z_n) + C\cos(y_n) \\ y_n + B\sin(x_{n+1}) + A\cos(z_n) \\ z_n + C\sin(y_{n+1}) + B\cos(x_{n+1}) \end{pmatrix}.$$

This is the so called ABC map. The example comes from fluid mechanics, and arises as a truncation of the time-τ map for the so called ABC flow. The ABC flow is a particular stationary solution of Euler equation, and the ABC map is a useful bridge between the study of 2 and 4 dimensional symplectic maps. (Notice that if we evaluate the components in xyz order, the expression does explicitly define a mapping on $\mathbb{R}^3$). The ABC map was introduced in [FKP88] in order to study chaos in the three dimensional volume preserving setting.

It is not hard to show that the phase space contains hyperbolic fixed points. Furthermore, it has been observed (at least empirically) that for many parameter values the ABC map admits chaotic motions [FKP88]. We use the tools developed in section 3.2 to compute a topological horseshoe factor about the fixed point p.

Take (somewhat arbitrarily) $A = 0.6543$, $B = 0.562$, $C = 0.701$, and consider the region $X = [-3, 7] \times [-1, 9] \times [-3, 7]$ with an initial resolution of $r = 0.125$. This gives an initial cubical complex containing 512,000 cubes. The large number of initial cubes illustrates the dependance of the complexity of set-oriented methods on the dimension. Note that this is an $80 \times 80 \times 80$ grid. In the plane the same resolution requires only 6,400 cubes, or roughly the same number of initial cubes as used in the standard map example.

Figure 4.9 shows the cubical fixed point set for this set of parameter values, as well as the local block $\mathcal{N}_0$ for one of the fixed points. We apply a homoclinic excursion search similar to algorithm 3.7 at the fixed point. A close-up of the initial local stable and unstable covers relative to this block are shown in figure 4.10.

The images of the low resolution globalized manifolds are hard to render effectively in three dimensions, (the eye makes some sense of the pictures if they are rotating, but still projections onto the page are very little help). Nevertheless, on the level of algorithms, the search proceeds as before. Figure 4.11 shows a cover of a homoclinic excursion which was obtained after five subdivisions.

5. Performance and Implementation. A C++ implementation of the computational scheme discussed in this work is available at [MJ]. All references below to timing, performance, and source code refer to this implementation.

5.1. Discretization of a C^2 Dynamical System. We implement a cubical complex as a vector of cubes, a cube as a vector of intervals, and an interval as a pair of double precision floating point numbers. Two intervals are the same if their end points agree to some specified precision, which we usually take to be 10^{-14}. Vector containers are implemented using the C++ standard template library class *vector* (for the STL see [PP00]). The source code for these elementary data structures is in the files *doubleCubical.h* and *doubleCubical.cc.*

To implement the discretization operator $\mathcal{F} = \textbf{enclose}(\mathcal{X}, f)$ it is necessary to compute, for each cube $Q \in \mathcal{X}$, a combinatorial image with $f(Q) \subset [\mathcal{F}(Q)]^\circ$. To compute the enclosure, we begin with the linear approximation of f by its derivative, and refine this approximation by bounding the nonlinearities. Assume then that $f \in C^2(\mathcal{X})$ (In the applications considered in this work, f is in fact real analytic).

Let $r > 0$ be the resolution of $\mathcal{X}$, and x_Q be the center of Q. For any $x \in Q$ we have, by Taylor's Theorem

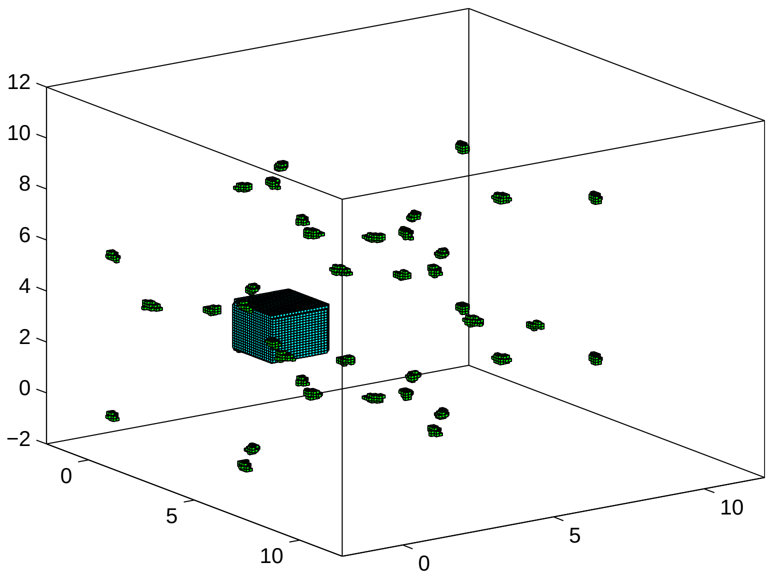

FIG. 4.9. *The local block $\mathcal{N}_0$ grown for a fixed cube.*

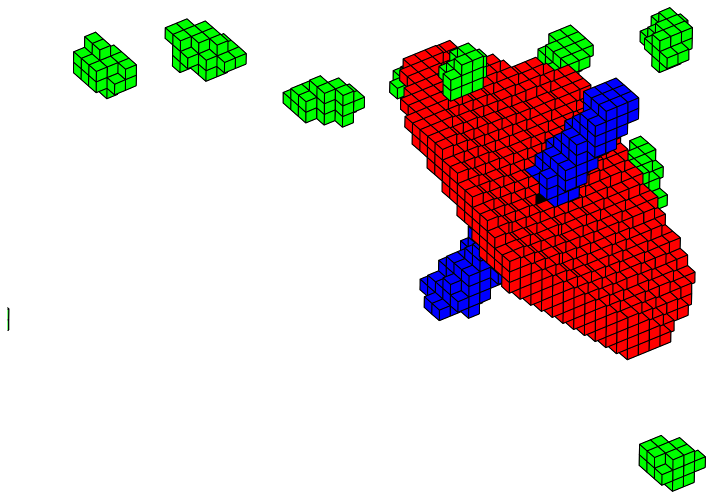

FIG. 4.10. *Initial coarse local stable and unstable covers about a fixed cubes.*

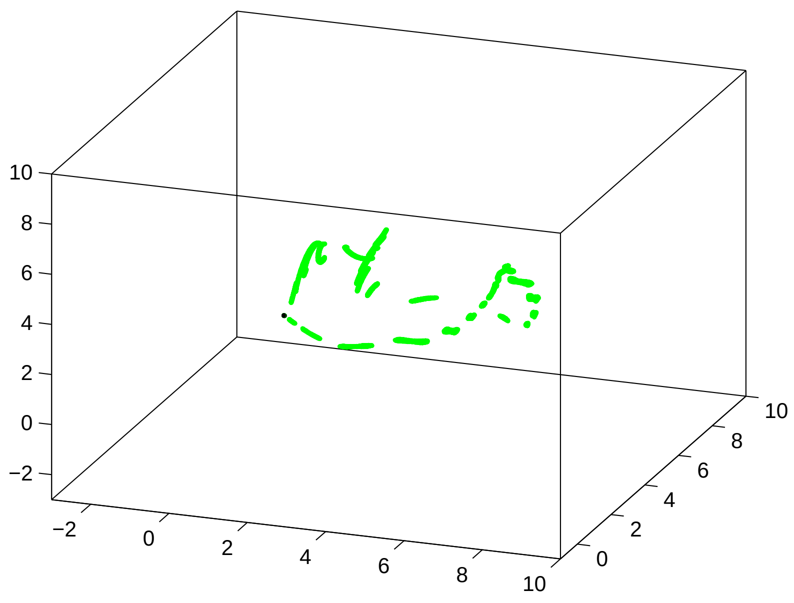

FIG. 4.11. *A cover of a homoclinic excursion for the ABC map. The fixed cube component can still be seen in black.*

$$f(x) = f(x_Q) + Df(x_Q) \cdot [x - x_Q] + R(x_Q, x - x_Q),$$

where the Taylor Remainder $R : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ can be computed explicitly, and has

$$\sup_{x \in Q} (\| R(x,r) \|) \leq \frac{1}{2} \| D^2 f \|_{C_0} \frac{nr^2}{4} \equiv \mathbf{R}(r)$$

We call $\mathbf{R}$ the truncation error estimate. Then $f(Q)$ is approximated by the parallelogram

$$f(Q) \approx P_Q \equiv f(x_Q) + Df(x_Q)[Q - x_Q],$$

and the truncation error in this approximation is explicitly known.

These observation form the basis for the following algorithms. The algorithms require that we have analytic expressions for f, Df, and the truncation error estimate $\mathbf{R}$.

ALGORITHM 5.1 (Compute Linear Image).

```
function linearImageOf ($Q, f, Df$);
$x_Q$ = center of $Q$;
$y = f(x_Q)$;
for (each $v_i$ a vertex of $Q$:);
        $e_i = v_i - x_Q$;
        $T_i = Df(x_Q)\, e_i$;
        $P_i = y + T_i$;
*end for*
*return* $\{P_i\}$;
```

Algorithm 5.2 (Compute an n-rectangle enclosing the image).

```
*function* computeEnclosure $(\mathcal{X}, Q, f, Df, \mathbf{R})$;
$n =$ **dimensionOf**$(\mathcal{X})$;
$r =$ **resolutionOf**$(\mathcal{X})$;
$x_Q =$ **centerOf**$(Q)$;
$\epsilon = |\mathbf{R}(x_Q, \sqrt{n}r/2)|$;
$P_i =$ **linearImageOf**$(Q, f, Df)$;
$P =$ **parallelogram**$(\{P_i\}_{i=1}^{2^n})$;
*for* $(1 \leq j \leq n)$;
        $I_j = \pi_j(P)$;
        $a =$ **leftEnd**$(I_j)$;
        $b =$ **rightEnd**$(I_j)$;
        $I'_j = [\ a - \epsilon, b + \epsilon\ ]$;
*end for*;
$B = I'_1 \times \ldots \times I'_n$;
$\mathcal{R}_Q =$ **include**$(\mathcal{X}, B)$;
*return* $\mathcal{R}_Q$;
```

Algorithm 5.3 (Minimize the Combinatorial Enclosure.).

```
*function* minimizeEnclosure $(\mathcal{X}, Q, f, Df, \mathbf{R})$;
$n =$ **dimensionOf**$(\mathcal{X})$;
$r =$ **resolutionOf**$(\mathcal{X})$;
$x_Q =$ **centerOf**$(Q)$;
$\epsilon = \mathbf{R}(x_Q, \sqrt{n}r/2)$;
$P_i =$ **linearImageOf**$(Q, f, Df)$;
$P =$ **parallelogram**$(\{P_i\}_{i=1}^{2^n})$;
$P' =$ **expand**$(P, \epsilon)$;
$\mathcal{R}_Q =$ **computeEnclosure**$(\mathcal{X}, Q, f, Df, \mathbf{R})$;
*for* (each $Q_j \in \mathcal{R}_Q$);
        *if* ($Q_j$ is outside $P'$);
                **delete**$(Q_j, \mathcal{R}_Q)$;
        *end if* ;
*end for*;
*return* $\mathcal{F}(Q) = \mathcal{R}_Q$;
```

To compute the combinatorial outer enclosure for a map f on a cubical domain $\mathcal{X}$, we call algorithm 5.3 for every $Q \in \mathcal{X}$. Note that algorithms 5.2 and 5.3 make use of some shared data. In practice the algorithms can be implemented so that this data is computed only once.

Algorithms 5.1 - 5.3 are implemented as member functions of a class *C2diffeo*. The maps f, Df, and the truncation estimator $\mathbf{R}$ are passed into the *C2diffeo* object as template parameters. (Passing the function definitions as template parameters forces the compiler to inline the function calls. See [Yan00], chapter 7.7). The source code for this data structure is found in the files *C2diffeomorphism.h*.

The class *parallelogram* is defined in the files *parallelogram.h* and *parallelogram.cc*. A parallelogram is implemented as a vector containing the center, a collection of

2^n vertices, and a collection of $2n$ faces. The class *face* holds a collection of 2^{n-1} references to the vertices of the parallelogram, as well as the outward unit normal vector to the face. The outward direction is defined by requiring that the center of the parallelogram has negative projection.

The call $P = \mathbf{parallelogram}(\{P_i\}_{i=1}^{2^n})$ instantiates a parallelogram P whose vertices are defined by the collection $\{P_i\}_{i=1}^{2^n}$. In this case the vertices are the images of the vertices of Q under the linear approximation. The function call $P' = \mathbf{expand}(P, \epsilon)$ returns a parallelogram P' enclosing the parallelogram P, and whose faces are a distance ϵ from, and parallel to the faces of P.

Expanding the parallelogram makes use of the outward unit normal directions of the faces. Similarly, in order to test if a grid cube $Q_j \in \mathbf{R}_Q$ is outside the parallelogram P', we test to see if there is a face F of P' so that all the vertices of Q_j are on one side of F. This is tested by choosing a point $w \in F$ and for each vertex $v_k \in Q_j$ testing the sign of the projection of $v_k - w$ onto the unit outward normal of F.

The functions π_j are the canonical projection operators, which in this case return the intervals onto which which P projects. The functions $\mathbf{leftEnd}(I)$, and $\mathbf{rightEnd}(I)$ return the left and right endpoints of an interval I respectively. The function $\mathbf{include}(\cdot, \cdot)$ is best thought of as a combinatorial outer enclosure of the identity map. It is a map which, given a cubical complex $\mathcal{X}$ and a set of points V, returns the minimal cubical sub-complex of $\mathcal{X}$ which covers the points in V. The **include** function is implemented as a member function of the *doubleCubicalSet* class. All parallelogram objects are in scope only during the execution of Algorithm 5.1.

For the maps used in this paper, where we have explicit expressions of f in terms of elementary functions, so that we can derive explicitly formulas for the differential and the error estimate. The standard map, for example, has differential

$$Df_\epsilon(x, \theta) = \begin{pmatrix} 1 & \epsilon\cos(\theta) \\ 1 & 1 + \epsilon\cos(\theta) \end{pmatrix},$$

and truncation error estimate

$$\mathbf{R}(x_Q, |h|) = 2\epsilon\left(|\cos(h_2) - 1| + |\sin(h_2) - h_2|\right).$$

Similar estimates can be worked out for all the examples used in this paper. The expressions for the differentials and the error estimators of all maps used throughout the present work, as well as the expressions of maps themselves are found in the file *discreteDynamicalSystems.h.*

Rather than employing interval arithmetic to account for round off error, the software over-estimates the number of floating point operations in the discretization process and adjusts the truncation error accordingly. For maps the computation of a combinatorial image involves only a few hundred floating point operations, so that it is reasonable to expect round off errors on the order of no more than 10^{-14}. In practice we simply double the (rigorous) truncation error estimate throughout the computation. In the examples in Section 4 the highest resolutions occurring in the computations are on the order of $r \approx 10^{-4}$ so that truncation errors are on the order of 10^{-8}, and doubling the truncation error estimate more than compensates for roundoff error. Nevertheless, the inclusion of interval arithmetic into the software could be easily accomplished by modifying only Algorithm 5.1.

5.2. Implementation of Combinatorial Data Structures. The combinatorial data $\mathcal{F}$ is stored as a vector of ordered sets of integers. For each $Q \in \mathcal{X}$ we store the index set $i_1, \ldots i_n$ where $\mathcal{F}(Q) = \{Q_{i_1}, \ldots, Q_{i_n}\}$. The sets $\{i_1, \ldots, i_n\}$ are implemented using the standard template library ordered set class. The source code for the combinatorial enclosure data structure is found in the files *dynamics.h* and *dynamics.cc.*

Once a system is discretized, the remaining combinatorial analysis is based on the operations of union, intersection, and difference for ordered sets of integers. Define the function **Index** : $\mathcal{X} \to \mathbb{N}^+$ by

$$\mathbf{Index}(Q_i) = i$$

where i is the vector index of a grid cube, and for $\mathcal{A} \subset \mathcal{X}$ let

$$\mathbf{Indices}(A) = \{i = \mathbf{Index}(Q_i) \mid Q_i \in A\}$$

be the function that returns the ordered set of indices of $\mathcal{A}$. If S is an ordered set of integers set of integers with $S \subset \mathbf{Indices}(\mathcal{X})$ let

$$\mathbf{Indices}^{-1}(S) = \{Q_i \in \mathcal{X} \mid i \in S\}$$

be the cubical sub-collection whose indices are the elements of S. We implement all of the set operations used in our algorithms on the index level. So for example when an algorithm calls for the computation of $\mathcal{A} \cup \mathcal{B}$, this is evaluated simply by computing the union $\mathbf{Indices}(\mathcal{A}) \cup \mathbf{Indices}(\mathcal{B})$.

For integer sets A and B with small cardinality we implement the set operations of union, intersection, and difference using standard template library algorithms. The complexity of the STL set operations is $N \log(N)$ with N the cardinality of the larger set. For sets with a large number of elements we implement the set operations using an axillary hash, in which case the complexity is be reduced to N. (This is a standard trick. See [CLRS01]).

5.3. Interface with CHomP and Graph Algorithms. The output of Algorithm 3.7 is a cubical set which presumably covers some homoclinic orbits. To rigorously verify the existence of a topological horseshoe, it is necessary to grow an index pair which is then processed by CHomP. While the Algorithm 2.9 is sufficient for computing the index pair, in practice we use a somewhat more involved procedure which grows an index pair which is pre-conditioned for the 'homcubes' program and which is minimal in some sense. The interested reader should consult the source code for the functions **indexPairSurgery**, and **growIndexPairNeighborhood**. These are defined in the header file *C2diffeomorphism.h.*

In order to process this index pair using CHomP, it is necessary to convert both the index pair $(\mathcal{P}_1, \mathcal{P}_0)$ and the combinatorial enclosure $\mathcal{F}$ into a form appropriate for input into CHomP. The reader interested in this interface can consult the source code for the member function **outToChomp** in the class 'combinatorialEnclosure'. The class and function are defined and implemented in the files *dynamics.h* and *dynamics.cc.*

One more comment on implementation. For several computations in appendix B we need to call Dijkstra's shortest path algorithm. To implement the dynamical graph we use the *Boost Graph Library* [JS], which provides templates for directed graph data structures as well as templates for the standard graph searches, including Dijkstra's Algorithm. Since we have already implemented the combinatorial enclosure data structure as a vector of ordered sets of integers, it is a trivial matter to convert

System	Total Run Time	Discretization Time	Set Operations Time
Henon Map	2.02 sec	1.08 sec (53.46%)	0.72 sec (35.64%)
Standard Map	9.92 sec	5.01 sec (50.5%)	4.91 sec (49.4%)
Suris Map	13.84 sec	9.65 sec (69.73%)	3.59 sec (25.94%)

Table 5.1
Performance of the transition chain searches.

System	Initial Grid Size	Initial h	Final Grid Size	Final h
Henon Map	2,500 cubes	0.28	558 cubes	0.004
Standard Map	6,400 cubes	0.15	4,703 cubes	0.005
Suris Map	10,000 cubes	0.03	2,470 cubes	0.002

Table 5.2
Initial and final complex data.

this data into a directed graph. The graph is implemented as an adjacency list, and the *Boost Graph Library* is equipped with constructors which facilitate this conversion. The source code for our implementation of the dynamical graph class is found in *dynamicalGraph.h* and *dynamicalGraph.cc.*

5.4. Performance. With these comments on implementation in place, we briefly discuss the performance of our algorithms. We only give the timing results for the discretization of the dynamical systems, and the connecting orbit search 3.7. We do not include performance data for the post processing and Conley computation stages, as these computations are well documented elsewhere in the literature.

Table 2 gives the timing data for the example computations in Section 4. The total program time, time spent in the discretization and set-oriented stages, as well as the percentage of time in each stage of the computation is shown for all four examples. The complexity of the computations depends strongly on the number of cubes in the domain. In order to tie the timing results to domain size, we give the initial and final grid sizes and resolutions in Table 3.

We remark that the difference between the initial resolution required in each of the examples is related to the dynamics of the given map. The fewest initial cubes are needed for the area preserving Henon map, while the Suris map requires the largest initial grid. The reason for this is that we chose to study the Henon map toward the anti-integrable limit, and to study the Suris map near the perturbative regime, while the parameter values for the standard map were taken between the integrable and anti-integrable limits.

6. Acknowledgements. The author would like to thank Rafael de la Llave and Konstantin Mischaikow for many useful conversations and encouragement as this work developed. Thanks also to Maorong Zou, Patrick Goetz, and Jeffery Strunk for substantial help with the computer resources of the UT math department. Thanks to Pawel Pilarczyk and J.T. Halbert for carefully reading an earlier draft of the manuscript and many helpful comments and corrections. Thanks to Hector Lomelí for suggesting the Suris map as an example of an a-priori unstable area preserving twist mapping, and to Phil Morrision for suggesting the ABC-mapping as an interesting example of a discrete volume preserving dynamical sy stem. The author was supported by NSF grant DMS 0354567 and by the University of Texas Department of Mathematics Program In Applied and Computational Analysis RTG Fellowship

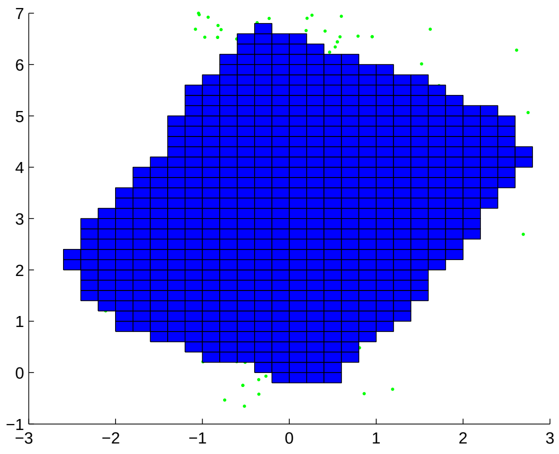

Fig. A.1. *Cubical cover of resonance island in the standard map. Computed using the invariant part algorithm.*

during this work.

Appendix A. Direct Application of Graph-Theoretic Methods to Conservative Maps..

In this Appendix we give three examples of how standard set oriented strategies fail for conservative systems. These examples illustrate explicitly the need for the approach considered in the body of the paper.

Example A.1.

One stumbling block for applying the methods of [KMV05] to conservative systems is the presence of chain recurrent subsets of large measure. Figure A.1 shows the results of applying algorithm 2.5 to the standard map, with a 50×50 *cubical covering of the domain* $[-3, 3] \times [-1, 7] \subset \mathbb{R}^2$*. The resulting invariant cubes are shown in blue.*

There is a topological horseshoe factor near the resonance, which we computed in Section 3.1. However, subdividing the chain recurrent set and recomputing the invariant part brings us no closer to isolating the horseshoe. Figure A.2 illustrate this. The figure shows the combinatorial invariant set after two additional subdivisions.

While the subdivision process culls cubes near the boundary of the set, it can never cull cubes in the interior of the resonance due to the existence of invariant circles. If we continue to subdivide this set, the number of cubes grows by roughly a factor of four at each step, but there is no improvement in the culling of the resonance. Compare this to the fast convergence of the subdivision scheme to the Henon attractor, as discussed in [DJM05] and [DFT08]. The presence of invariant tori in volume preserving maps makes this problem even more pronounced in three dimensions.

Another possible strategy is the direct computation of graph theoretic search procedures to the chain recurrent subset containing the resonance zone. The next two examples illustrate the fact that graph searches are especially unreliable for conservative systems.

Example A.2.

In this example we attempt to compute a homoclinic excursion for the hyperbolic fixed point at the origin in the standard map, using Dijkstra's Algorithm. We take the

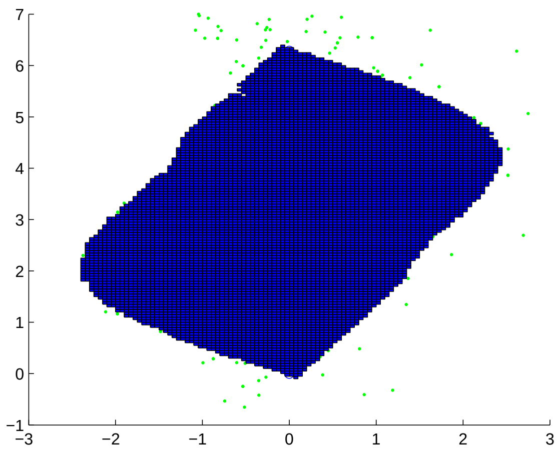

FIG. A.2. *Result of applying the top-down strategy for dissipative maps to a chain recurrent subset of the standard map. The figure shows the resulting invariant cubes after two subdivisions. Number of cubes in cover grows geometrically.*

same 50×50 *grid of* 2,500 *cubes as in example A.1. Denote the output of Dijkstra's shortest path algorithm by* $\mathcal{SP}$. *The result of the search is shown in Fig A.3.*

The resulting combinatorial excursion is a false positive, as can be either verified graphically (by noting that the orbit passes from the interior to the exterior of an invariant circle), or by subdividing the connection and re-computing the invariant part. If this is done, the connection disappears, and it is clear that the shortest path algorithm has failed to locate a true homoclinic excursion.

It could be argued that we have chosen too poor an initial resolution, however this is the same initial grid used in section 4.1 *to compute a horseshoe factor for the standard map. The point of the example is that while the initial grid may be too coarse for the straight forward application of graph-theoretic algorithms, the grid is not so coarse that we cannot proceed by more geometrical means.*

EXAMPLE A.3.

A final example highlights problems associated with the abundance of recurrence in measure preserving systems. The abundance of recurrent orbits tends to make the dynamical graph associated with a conservative system strongly connected, regardless of resolution. Then we subdivide the domain from example A.2 twice, and obtain a cubical grid composed of 40,000 *cubes. Applying the Dijkstra's Shortest Path algorithm to the elliptic and hyperbolic fixed points gives the result shown in Fig A.4. The red cubes in the center and the blue cubes at the bottom of the figure are the combinatorial fixed cubes, covering neighborhoods of the elliptic and hyperbolic fixed points respectively. The black cubes show a combinatorial shortest path from the hyperbolic, to the elliptic fixed point. It is obvious that while no such orbit is possible in the underlying dynamics, we must expect an abundance of such false positives near rotational dynamics in conservative systems.*

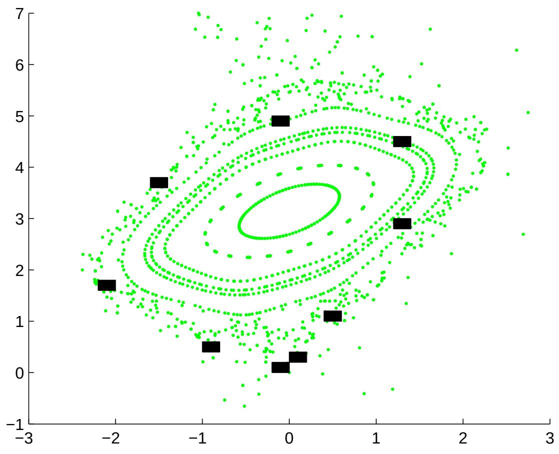

FIG. A.3. *Dijkstra shortest path from the fixed cube back to itself (after the self path has been disallowed).*

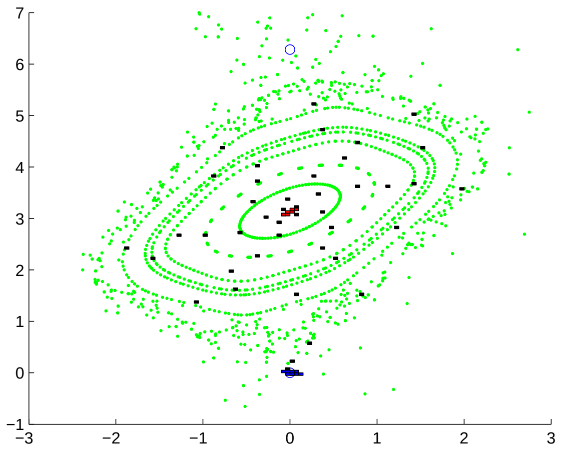

FIG. A.4. *Fixed cubes shown red and blue. Black cubes are Dijkstra's shortest path from fixed set to fixed set.*

REFERENCES

[AKK$^+$09] Z Arai, W Kalies, H Kokubu, K Mischaikow, H Oka, and P Pilarczyk. A database schema for the analysis of global dynamics of multiparameter systems. *SIAM Journal on Applied Dynamical Systems*, 8(3):757–789, 2009.

[BW95] Keith Burns and Howard Weiss. A geometric criterion for positive topological entropy. *Comm. Math. Phys.*, 172(1):95–118, 1995.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, second edition, 2001.

[DFT08] Sarah Day, Rafael Frongillo, and Rodrigo Treviño. Algorithms for rigorous entropy bounds and symbolic dynamics. *SIAM J. Appl. Dyn. Syst.*, 7(4):1477–1506, 2008.

[DH96] Michael Dellnitz and Andreas Hohmann. The computation of unstable manifolds using subdivision and continuation. In *Nonlinear dynamical systems and chaos (Groningen, 1995)*, volume 19 of *Progr. Nonlinear Differential Equations Appl.*, pages 449–459. Birkhäuser, Basel, 1996.

[DH97a] Michael Dellnitz and Andreas Hohmann. A subdivision algorithm for the computation of unstable manifolds and global attractors. *Numer. Math.*, 75(3):293–317, 1997.

[DH97b] Michael Dellnitz and Andreas Hohmann. A subdivision algorithm for the computation of unstable manifolds and global attractors. *Numer. Math.*, 75(3):293–317, 1997.

[DJK$^+$05] Michael Dellnitz, Oliver Junge, Wang Sang Koon, Francois Lekien, Martin W. Lo, Jerrold E. Marsden, Kathrin Padberg, Robert Preis, Shane D. Ross, and Bianca Thiere. Transport in dynamical astronomy and multibody problems. *Internat. J. Bifur. Chaos Appl. Sci. Engrg.*, 15(3):699–727, 2005.

[DJM04a] S. Day, O. Junge, and K. Mischaikow. A rigorous numerical method for the global analysis of infinite-dimensional discrete dynamical systems. *SIAM J. Appl. Dyn. Syst.*, 3(2):117–160 (electronic), 2004.

[DJM04b] S. Day, O. Junge, and K. Mischaikow. A rigorous numerical method for the global analysis of infinite-dimensional discrete dynamical systems. *SIAM J. Appl. Dyn. Syst.*, 3(2):117–160 (electronic), 2004.

[DJM05] Sarah Day, Oliver Junge, and Konstantin Mischaikow. Towards automated chaos verification. In *EQUADIFF 2003*, pages 157–162. World Sci. Publ., Hackensack, NJ, 2005.

[DJPT06] Michael Dellnitz, Oliver Junge, Marcus Post, and Bianca Thiere. On target for Venus—set oriented computation of energy efficient low thrust trajectories. *Celestial Mech. Dynam. Astronom.*, 95(1-4):357–370, 2006.

[DJT01] M. Dellnitz, O. Junge, and B. Thiere. The numerical detection of connecting orbits. *Discrete Contin. Dyn. Syst. Ser. B*, 1(1):125–135, 2001.

[FKP88] Mario Feingold, Leo P. Kadanoff, and Oreste Piro. Passive scalars, three-dimensional volume-preserving maps, and chaos. *J. Statist. Phys.*, 50(3-4):529–565, 1988.

[GdlL06] Marian Gidea and Rafael de la Llave. Topological methods in the instability problem of Hamiltonian systems. *Discrete Contin. Dyn. Syst.*, 14(2):295–328, 2006.

[GJ05] Lars Grüne and Oliver Junge. A set oriented approach to optimal feedback stabilization. *Systems Control Lett.*, 54(2):169–180, 2005.

[GR04] Marian Gidea and Clark Robinson. Symbolic dynamics for transition tori. II. In *New advances in celestial mechanics and Hamiltonian systems*, pages 95–108. Kluwer/Plenum, New York, 2004.

[HB06] W. Kalies H. Ban. A computational approach to conley's decomposition theorem. *Journal of Computational and Nonlinear Dynamics 1*, pages 312–319, 2006.

[HMZ88] M.-R. Herman, R. McGehee, and E. Zehnder, editors. *Charles Conley memorial volume.* Cambridge University Press, Cambridge, 1988. Special Issue of Ergodic Theory and Dynamical Systems. Vol. 8*.

[JS] A. Lumsdaine J. Siek, L Lee. *The Boost Graph Library.* User Guide and Reference Manual.

[Jun01] Oliver Junge. An adaptive subdivision technique for the approximation of attractors and invariant measures: proof of convergence. *Dyn. Syst.*, 16(3):213–222, 2001.

[Kat] Anatole Katok. *Introduction to the modern theory of dynamical systems*, volume 54 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press. With a supplementary chapter by Katok and Leonardo Mendoza.

[KMM04] Tomasz Kaczynski, Konstantin Mischaikow, and Marian Mrozek. *Computational homology*, volume 157 of *Applied Mathematical Sciences.* Springer-Verlag, New York, 2004.

[KMV05] W. D. Kalies, K. Mischaikow, and R. C. A. M. VanderVorst. An algorithmic approach to chain recurrence. *Found. Comput. Math.*, 5(4):409–449, 2005.

[MG08] C. Robinson M. Gidea. Obstruction argument for transition chains of tori interspersed with gaps. *Continuous Dynamical Systems*, Submitted, 2008.

[Mis99] Konstantin Mischaikow. The Conley index theory: a brief introduction. In *Conley index theory (Warsaw, 1997)*, volume 47 of *Banach Center Publ.*, pages 9–19. Polish Acad. Sci., Warsaw, 1999.

[MJ] J.D. Mireles James. ACVLCM++: Automated chaos verrification library for conservative maps (dynamical systems software in c++). *www.math.utexas.edu/users/jjames/codes/ACVLCM++.*

[MM98] Konstantin Mischaikow and Marian Mrozek. Chaos in the Lorenz equations: a computer assisted proof. II. Details. *Math. Comp.*, 67(223):1023–1046, 1998.

[MMP05] Konstantin Mischaikow, Marian Mrozek, and Pawel Pilarczyk. Graph approach to the computation of the homology of continuous maps. *Found. Comput. Math.*, 5(2):199–229, 2005.

[Osi07] George Osipenko. *Dynamical systems, graphs, and algorithms*, volume 1889 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2007. Appendix A by N. B. Ampilova and Appendix B by Danny Fundinger.

[PdM82] Jacob Palis, Jr. and Welington de Melo. *Geometric theory of dynamical systems*. Springer-Verlag, New York, 1982. An introduction, Translated from the Portuguese by A. K. Manning.

[Pil97] Paweł Pilarczyk. *The CHomP Software, in Computational Homology Project.* httm://chomp.rutgers.edu/, 1997.

[PP00] M. Lee F. Musser P. Plauger, A. Stepanov. *The C++ Standard Template Library.* Perntice Hall PTR, 2000. User Guide and Reference Manual.

[PS08] Paweł Pilarczyk and Kinga Stolot. Excision-preserving cubical approach to the algorithmic computation of the discrete Conley index. *Topology Appl.*, 155(10):1149–1162, 2008.

[RdlL] M. Gidea R. de la Llave. Arnold diffusion with optimal time in the large gap problem. *Preprint.*

[Rob02] Clark Robinson. Symbolic dynamics for transition tori. In *Celestial mechanics (Evanston, IL, 1999)*, volume 292 of *Contemp. Math.*, pages 199–208. Amer. Math. Soc., Providence, RI, 2002.

[Sma65] Stephen Smale. Diffeomorphisms with many periodic points. In *Differential and Combinatorial Topology (A Symposium in Honor of Marston Morse)*, pages 63–80. Princeton Univ. Press, Princeton, N.J., 1965.

[Srz00] Roman Srzednicki. A generalization of the Lefschetz fixed point theorem and detection of chaos. *Proc. Amer. Math. Soc.*, 128(4):1231–1239, 2000.

[Sur94] Yu. B. Suris. A discrete-time Garnier system. *Phys. Lett. A*, 189(4):281–289, 1994.

[Szy95] A. Szymczak. The Conley index for discrete semidynamical systems. *Topology Appl.*, 66(3):215–240, 1995.

[Szy97] Andrzej Szymczak. A combinatorial procedure for finding isolating neighbourhoods and index pairs. *Proc. Roy. Soc. Edinburgh Sect. A*, 127(5):1075–1088, 1997.

[Yan00] Daoqi Yang. *C++ and Object-oriented Numeric Computing for Scientists and Engineers.* Springer, 2000.