Artificial Intelligence 51 (1991) 179–221
Elsevier

Understanding complex dynamics by visual and symbolic reasoning

Kenneth Man-Kam Yip

Department of Computer Science, Yale University, P.O. Box 2158, Yale Station, New Haven, CT 06520, USA

Abstract

Yip, K.M.-K., Understanding complex dynamics by visual and symbolic reasoning, Artificial Intelligence 51 (1991) 179–221.

Professional scientists and engineers routinely use nonverbal reasoning processes and graphical representations to organize their thoughts and as part of the process of solving otherwise verbally presented problems. This paper presents a computational theory and an implemented system that capture some aspects of this style of reasoning. The system, consisting of a suite of computer programs collectively known as KAM, uses numerical methods as a means to shift back and forth between symbolic and geometric methods of reasoning. The KAM program has three novel features: (1) it articulates the idea that "visual mechanisms are useful for problem solving" into a workable computational theory, (2) it applies the approach to a domain of great technical difficulty, the field of complex nonlinear chaotic dynamics, and (3) it demonstrates the power of the approach by solving problems of real interest to working scientists and engineers.

1. Introduction

In observing professional physicists and engineers, we are often struck by how an expert's "intuitive grasp" of a field is hard to articulate verbally. This is perhaps indicative of the use of nonverbal reasoning processes as part of the process of solving otherwise verbally presented problems. We observe scientists, mathematicians, and engineers continually using graphical representations to organize their thoughts about a problem.

This paper presents a computational theory and an implemented system that capture some aspects of this style of reasoning. The system I developed—consisting of a suite of computer programs collectively known as KAM—uses numerical methods as a means of shifting back and forth between symbolic and geometric methods of reasoning. These programs not only draw graphs and diagrams, but look at these diagrams and hold them in their "mind's

eye" so that a powerful visual mechanism can be brought to bear on what otherwise would be purely symbolic problems. This style of reasoning is an example of what I call *imagistic reasoning*.

The idea that problem solvers employing visual, analogue, or diagrammatic representations can be more effective than those relying on linguistic representations alone is not new. Even before 1960, Gelernter's [3] Geometry Theorem Proving Machine used diagrams to filter goals generated by backward chaining. Nevins' [12] forward-chaining theorem prover focused its forward deduction of facts on lines explicitly drawn in a diagram. Stallman and Sussman's [16] EL program performed antecedent deductions by exploiting the finite connectivity of devices. Finally, Novak's [13] ISAAC program used diagrams to solve word problems in physics.

What is new in the KAM program, however, is the demonstration that this classic AI problem-solving paradigm can be made to work in a significant way. Specifically the research reported here contributes to the field of qualitative physics in three ways:

- It articulated the idea that "visual mechanisms are useful for problem solving", transforming it into a workable theory.
- It applied the theory to an area of great technical difficulty—complex nonlinear chaotic dynamics; this is an area that has never before been addressed by the AI community.
- It demonstrated that the approach is powerful enough to solve not just toy problems, but problems of real interest to working scientists and engineers. In one instance, KAM has led to previously unknown publishable results in hydrodynamics [19,23, Chapter 6].

The KAM program is concerned with the automatic qualitative analysis of nonlinear Hamiltonian systems with two degrees of freedom. Such systems arise from the equations of motions of particles in conservative fields. They are of essence in dynamical astrophysics, in plasma physics, and in accelerator design; a great deal of human time is spent in exploring their behavior. Although computers can painlessly generate particular numerical solutions to such equations, understanding the qualitative content of the equations requires substantial human effort and judgement to develop only relevant numerical simulations and to interpret the numerical results in high-level, qualitative terms.

KAM performs numerical simulations of Hamiltonian systems governed by their equations of motions. It automatically plans the numerical experiments, monitors its progress, and summarizes the numerical results in an executive summary containing information essential for dynamicists to understand the behavior of the systems. Performing the numerical simulations themselves, of course, is straightforward. The trick is to decide which sets of system parameters to try, how long to run any particular simulations, and

when one has gathered enough information to deduce the essential behavior of the system.

Typically, when an investigator encounters such a system, he attempts to characterize it with a consistent family of phase portraits. For each assignment of values of the system parameters, there is a phase portrait that shows the classes of trajectories that are possible for different initial conditions. Such a family of phase portraits can be obtained by running an immense number of expensive grid of initial conditions and system parameters, but an intelligent explorer can obtain the essential information with many fewer experiments. The reason is that although nonlinear dynamical behavior is complicated, there is structure on phase space that restricts the class of legal trajectories in one phase portrait, and provides a "grammar" of legal phase portraits.

This research work is rooted in the tradition of focusing on the problem-solving behavior of articulate professionals in well-structured domains and formalizing their methods so that a computer can exhibit similar behavior on similar problems. The KAM program captures the knowledge and judgement ordinarily supplied by a human investigator in conducting selective numerical experiments. It borrows techniques from many areas of computer science and artificial intelligence. For example, it uses numerical methods to evolve the trajectories and draws them in its mind's eye. It uses algorithms from computer vision and computational geometry to classify and recognize the relevant geometrical properties of trajectories. It uses classic AI techniques similar to Waltz's constraint analysis [20] to implement structure theorems that underly the "grammar" of dynamical behavior used to constrain the search and to produce summary reports.

The paper is organized as follows. We first describe what it is like to interact with KAM, illustrating its capabilities. We next show the major pieces of KAM and its underlying computational structure. We then take a closer look at the technology behind the demonstration results. We finally explain why KAM works and summarize the major contributions of this research.

2. A session with KAM

The following sample dialogue shows how KAM aids a scientist in exploring the dynamics of the Henon map [5]. KAM is written in Zetalisp, running on the Symbolics Lisp Machines. It is completely working; all the capabilities shown have been implemented and tested on many examples. The session is presented in part via an English dialogue solely for the benefit of the reader. KAM has no natural language capability; I have simply paraphrased KAM's output in stylized English.

Scientist. Here are the equations for Henon map, the definition of phase space, and the range of parameter value. [*He types in the equations (Fig. 1). He also types in range of state variables and the parameter.*] Consider the phase portrait with $\alpha = 1.3284305$. Show me the orbit starting at the initial state as indicated by the mouse click. [*The initial state is (0.08, 0.72).*]

KAM. The orbit is a QUASIPERIODIC ORBIT with rotation number between 5/26 and 1/5. It took 256 iterates to reach steady state (Fig. 2(a)).

Scientist. Do the same for the new mouse click. [*The initial state is (0.046, 0.011).*]

KAM. The orbit is a QUASIPERIODIC ORBIT with rotation number between 4/19 and 3/14. It took 256 iterates to reach steady state (Fig. 2(b)).

Scientist. Show me the entire phase portrait.

KAM. [*KAM decides what initial states to look at. After some time, KAM displays a phase portrait (Fig. 3), and reports its findings.*] The portrait has an elliptic fixed point at (0.000, 0.000). Surrounding the fixed point is a regular region bounded by a KAM curve with rotation number between 1/5 and 1/4. Outside the regular region lies a chain of five islands. The island chain is bounded by a KAM curve with rotation number between 4/21 and 5/26. The outermost region is occupied by ESCAPE orbits. Chaotic orbits occupy 62% of phase space.

Scientist. Show me all the major bifurcations as the parameter α is varied.

KAM: [*KAM decides what parameter values to look at. After some time, KAM makes the following executive report.*] In the parameter range (0, 2.2), KAM finds a total of 38 phase portraits. The following bifurcation patterns are observed:

- Poincaré–Birkhoff bifurcation: A pair of period-8 orbits appears at about 0.875 and the 8-island chain disappears at about 1.075.
- Poincaré–Birkhoff bifurcation: A pair of period-7 orbits appears at about 1.000 and the 7-island chain disappears at about 1.157.
- Poincaré–Birkhoff bifurcation: A pair of period-6 orbits appears at about 1.075 and the 6-island chain disappears at about 1.257.

HENON MAP

$$x_{n+1} = x_n \cos\alpha - (y_n - x_n^2)\sin\alpha$$
$$y_{n+1} = x_n \sin\alpha + (y_n - x_n^2)\cos\alpha$$

Range of x: (−1 1)
Range of y: (−1 1)
Range of α: (0 2.2)
Escape range: (−2 2)

Fig. 1. Input to KAM: (1) equations of motion, (2) range of state variables, (3) range of parameter, and (4) escape range.

- Poincaré–Birkhoff bifurcation: A pair of period-5 orbits appears at about 1.271 and the 5-island chain disappears at about 1.437.
- Poincaré–Birkhoff bifurcation: A pair of period-4 orbits appears at about 1.574 and the 4-island chain disappears at about 1.655.
- Poincaré–Birkhoff bifurcation: A pair of period-7 orbits appears at about 1.819 and the 7-island chain disappears at about 1.873.
- Extremal bifurcation: A pair of period-3 orbits appears at about 2.003 and the 3-island chain disappears at about 2.037.
- Phantom-3-kiss bifurcation: The unstable period-3 orbit collides with fixed point at about 2.064.

We have just seen what it is like to interact with KAM. As shown in the dialogue, KAM has the following capabilities:

- It decides when an orbit reaches steady state.
- It identifies the type of an orbit.
- It decides what initial states to look at.
- It decides what parameter values to look at.

In addition, KAM is able to interact with the user at a high conceptual level. It talks about fixed point, orbit, rotation number, regular and chaotic region, and bifurcation. It generates a textual analysis that explains what it sees. It provides the dynamicist with information that he really needs to understand the essential character of the system.

3. KAM overview: its structure and processes

This section describes the design criteria of KAM, its major pieces, and the computational processes responsible for deriving a high-level symbolic description from collections of discrete points in the phase space. We will

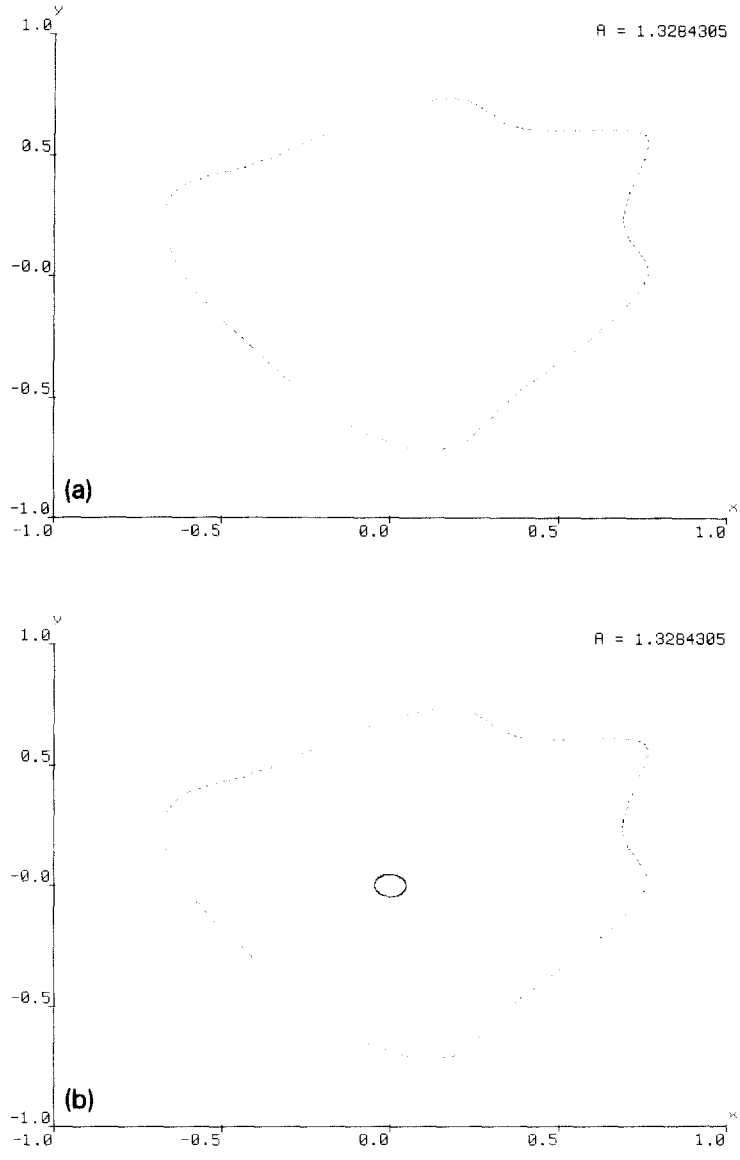


Fig. 2. KAM decides how many iterates to look at. It also identifies the type of an orbit. If the orbit is quasiperiodic, KAM computes a bound for its rotation number.

see that KAM has a very simple computational structure: it is made up of three layers of interpreters and each layer has the *same* pattern of behavior.

3.1. Design criteria

Three criteria govern the design of KAM:

- *The decisions that KAM makes must be high quality.* That means each decision to look for an orbit or a phase portrait must have

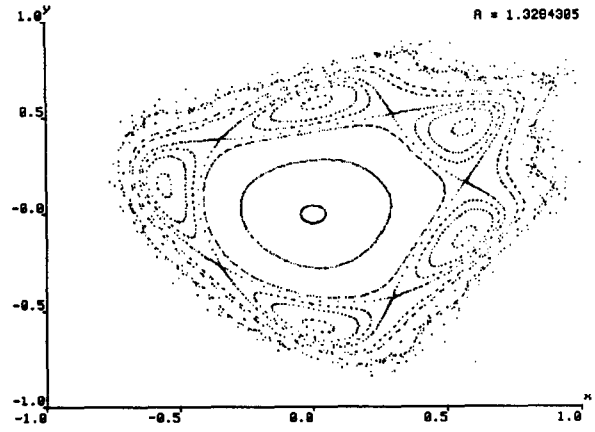


Fig. 3. KAM decides what initial states to pick. The phase portrait it produces consists of 10 orbits.

a high probability of finding something interesting. The quality of decision can be measured in terms of the number of orbits and phase portraits that KAM has to try in order to describe a mapping's behavior globally.

- *The descriptions of behavior that KAM gives must be concise, manipulatable by other programs, and useful to a dynamicist.* That means KAM has to be a useful tool for someone doing real science, and should be organized in a modular way that allows future expansion of its capabilities.
- *The consequences of KAM's execution must be understandable in terms of its knowledge of geometric algorithms and Hamiltonian dynamics.* That means two things: first, the knowledge that KAM has must be intelligible to a dynamicist, and hence is sharable and teachable to a novice; second, KAM's behavior can be explained in terms of knowledge that is independent of the particular program implementation.

3.2. Major components of KAM: orbit recognition, phase space searching, and parameter space searching

Figure 4 shows the basic components of KAM. There are four computer programs: (1) the window interface, (2) the orbit recognition program, (3) the phase space searching program, and (4) the parameter space searching program.

KAM operates in the following manner. The user interacts with the window interface. He can invoke different programs of KAM via the commands in the interface. The user input consists of data about the mapping that he wants to explore. The orbit recognition program conducts automatic or-

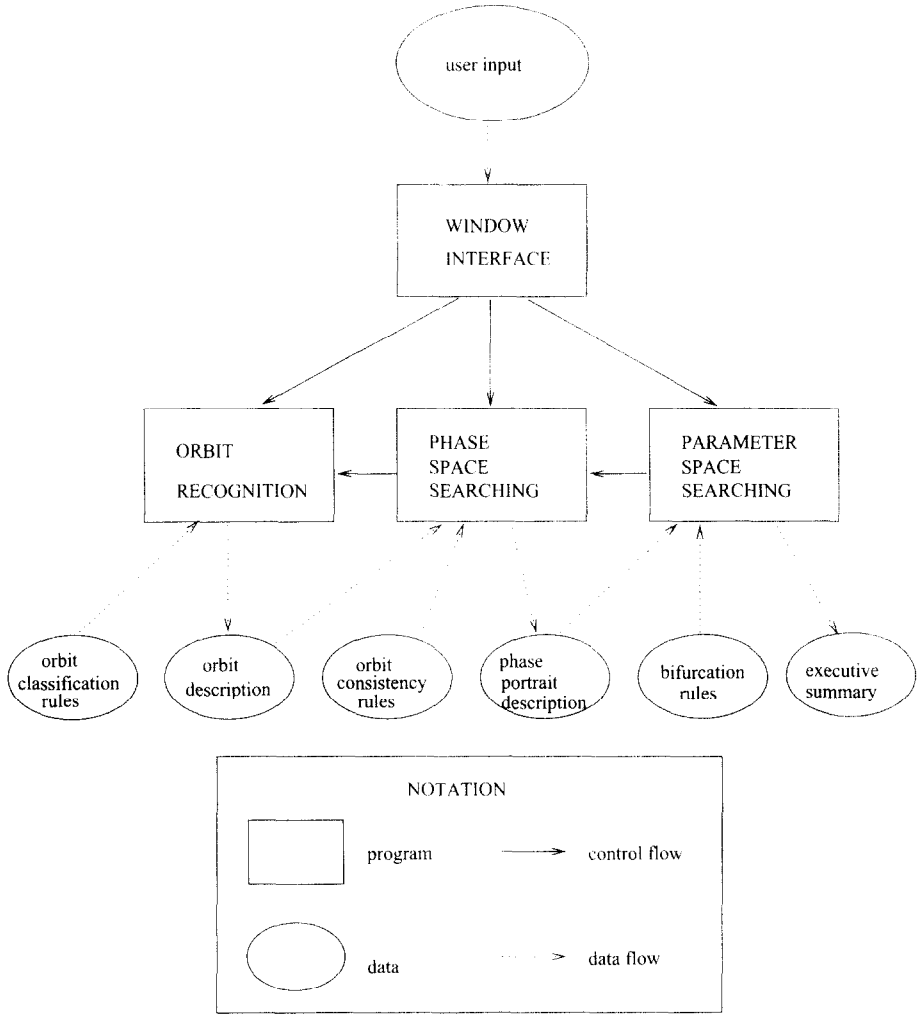


Fig. 4. Major components of KAM. The three major pieces are the orbit recognition module, phase space searching module, and the parameter space searching module.

bit recognition. It requires a knowledge base of known orbit types. Its output is a symbolic description of the orbit. The phase space searching program finds representative orbits in phase space. It requires a set of orbit consistency rules, which operates on the symbolic descriptions of orbits. Its output is a symbolic description of the phase portrait. The last component, the parameter space searching program, searches for all interesting bifurcations as a system parameter is varied. It requires a list of bifurcation rules, which operates on the symbolic descriptions of phase portraits. Its output is an executive report summarizing the major behaviors of the mapping.

3.3. Computational processes: aggregation, partition, and classification

The goal of KAM is to extract useful, high-level information about the behaviors of a mapping from a collection of point sets representing orbits in phase space. In solving this problem, KAM faces two central questions:

(1) where should it look for these point sets, and
(2) how can a symbolic orbit or phase portrait or bifurcation description be arrived at?

As there is a large semantic gap between the input (which is sets of discrete points) and the output (which denotes some meaningful events in Hamiltonian dynamics), it is almost certainly impossible to transform the input into the output in one step. A more reasonable strategy is to proceed via a sequence of intermediate representations that allow the gradual recovery of shape properties, spatial relations, and eventually the more global, dynamical properties of the system. It thus becomes important for KAM to view an object at different levels of representations such as points on the plane, a curve, an orbit, part of a flow pattern, and so on. What is needed is a theory of multiple representations and algorithms for going from one level of representation to another.

Figure 5 displays the structure of the computational processes of KAM. Abstracting away the specifics of Hamiltonian dynamics, the computational structure of KAM can be described as follows. The computation of spatial relations and recognition of shapes are organized into three different levels. Each level is equipped with a domain-independent geometry (for example, the dimension and topology of the ambient space, and an appropriate distance metric), and a list of domain-specific semantic labeling rules. Within each level, properties and relations that are not explicitly represented in the initial representation are extracted by applying three operations sequentially:

(1) aggregating isolated objects by some adjacency relation,
(2) partitioning the whole aggregate into meaningful subparts, and
(3) assigning a semantic label to the structured object according to domain-specific classification rules.

Semantically labeled objects of one level are then promoted to a higher level where the three operations are applied again but with different grouping and partitioning criteria.

Let me elaborate on the three semantic levels.

Orbit level

The bottom level—the orbit level—concerns orbit recognition. Its purpose is to assign a semantic category, an orbit type, to a collection of points based

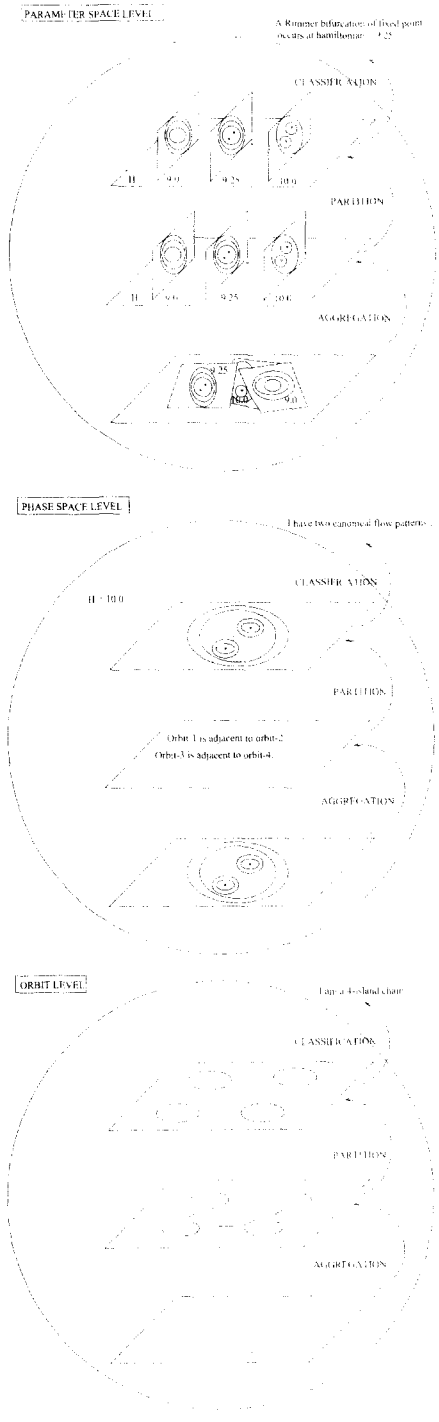


Fig. 5. Computational Structure of KAM. Focus on the shape of the processes. The same pattern of behavior—aggregation, partition, and classification—occurs in each of the three semantic levels.

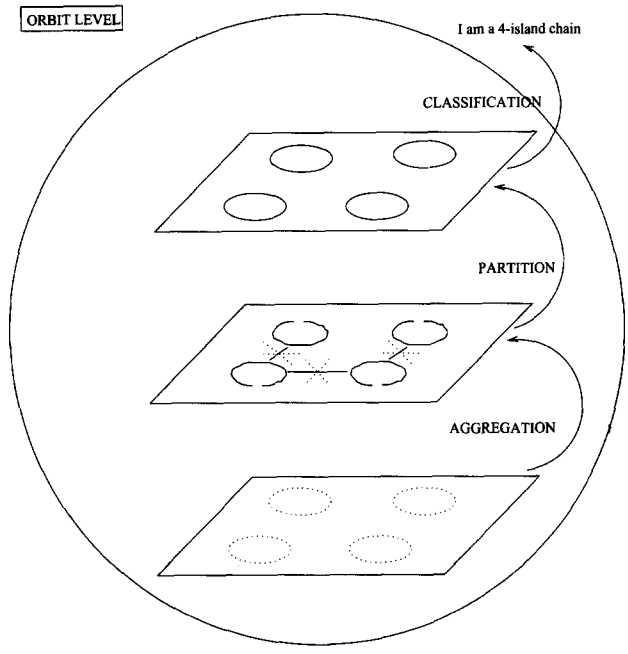


Fig. 6. The collection of isolated points are aggregated in a MST. The MST is partitioned into clusters by deleting inconsistent edges. The inconsistent edges are marked by the crosses (×). The geometry and grouping structure of the point set are described symbolically. A point set is classified by matching its symbolic description with the symbolic descriptions of known orbit types.

on the geometry of the point set. The recognition process proceeds in three steps (Fig. 6):

(1) aggregating points into a minimal spanning tree
(2) partitioning the minimal spanning tree into clusters
(3) classifying the orbit

Aggregation. Iterating a mapping on a given initial point generates a sequence of points. The first step in recognition is to aggregate these isolated points into a larger structure, the minimal spanning tree (MST). The MST defines an adjacency relation between two points: namely, two points are adjacent if and only if there is an edge connecting them in the MST representation. Certain shape information can already be extracted from the MST. For example, if the points lie on a closed curve, its MST representation will not have any nodes that have degree three or higher.

Partition. The second step is to detect clusters in the point set. This step is crucial because it tells us the clustering structure of the orbit. For example, to recognize a 4-island chain, it is essential to find out that the point set consists

of four distinct clusters. To find the clusters, KAM looks for inconsistent edges—edges that are significantly longer than their nearby edges—in the MST. Deleting the inconsistent edges breaks up the MST into subtrees, each of which represents a cluster. The output of the partitioning step is a symbolic description of the point set based on its geometry and clustering structure. For instance, in Fig. 6, the point set is described as an object with four parts, and each part is a closed curve.

Classification. The final step is to assign an orbit type to the point set. The classification step begins by matching the symbolic description of the point set against a list of classification rules. If the description matches a known orbit type, then the type of orbit is successfully identified. The orbit is passed to the next level. Otherwise, KAM assumes the input point set does not contain enough points to reveal the entire structure of the orbit. So it requests more points, repeating the orbit recognition process.

Phase space level

The middle level—the phase space level—concerns finding a set of orbits that are representative, in the sense that they completely characterize the global, qualitative behavior of the mapping. This search process is known as *phase space searching.* Its input is a collection of orbits whose types are already known. The output is a self-consistent phase portrait, a collection of orbits that fit together in a consistent way. Phase space searching consists of three steps (Fig. 7):

(1) aggregating orbits;
(2) partitioning orbits into groups;
(3) classifying each group according to canonical flow patterns.

Aggregation. The first step of the search process is to determine the adjacency relations among the orbits. The collection of orbits are aggregated into a data structure, an orbit adjacency network. The aggregation criterion here is different from that of the orbit level. Two orbits are adjacent if it is possible to draw a straight line between them without crossing a third orbit. The aggregation step is crucial because it guides KAM to search for representative orbits in the phase space. Specifically, KAM uses a list of domain-specific consistency rules to check whether two neighboring orbits are pairwise consistent. If they are not, then there must be some missing orbits between them. The output of the aggregation step is a self-consistent orbit adjacency network.

Partition. The second step is to determine the grouping structure of the orbits. It is a domain-specific fact that the collection of orbits surrounding a fixed point or a periodic orbit is an important structure to name and

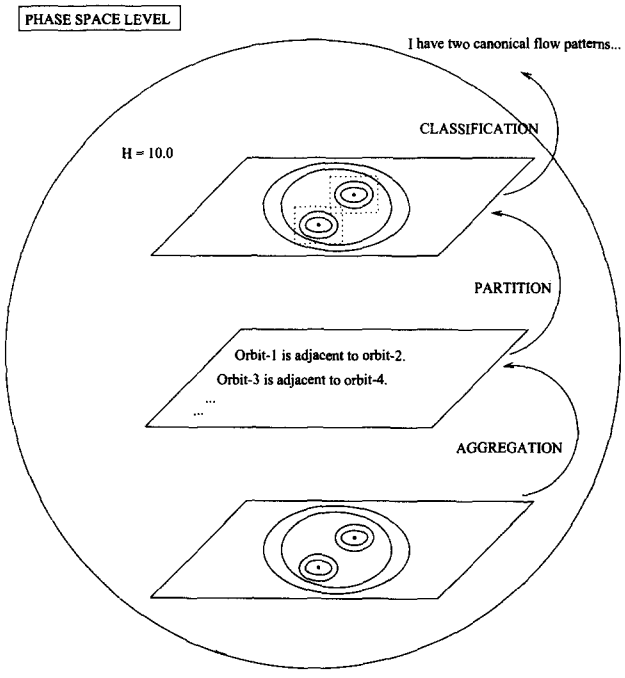


Fig. 7. The collection of orbits are aggregated in an orbit adjacency network. The spatial relationships among orbits are symbolically described. The symbolic description is the basis for grouping orbits into canonical flow patterns. Each dotted box encloses a canonical flow pattern.

classify. To find the groupings, KAM determines which orbit is inside or outside which other orbit. The output of the partition step is a symbolic description of the local structure around a fixed point or a primary periodic orbit.

Classification. The final step is to assign a semantic category to each local structure. The categories are called *canonical flow patterns*, which are equivalence classes of sets of orbits based on the topological structure of the orbits around a primary orbit. The output of the classification step is a phase portrait consisting of several canonical flow patterns. The phase portrait is then passed to the top level.

Parameter space level

So far we have talked about objects—orbits and collections of orbits—within a single picture (one phase portrait). Now, we turn our attention to relationships among different pictures.

The top level—the parameter space level—concerns finding all the interesting bifurcations as some parameter of the mapping is varied. The search process is called *parameter space searching*. A bifurcation corresponds to a

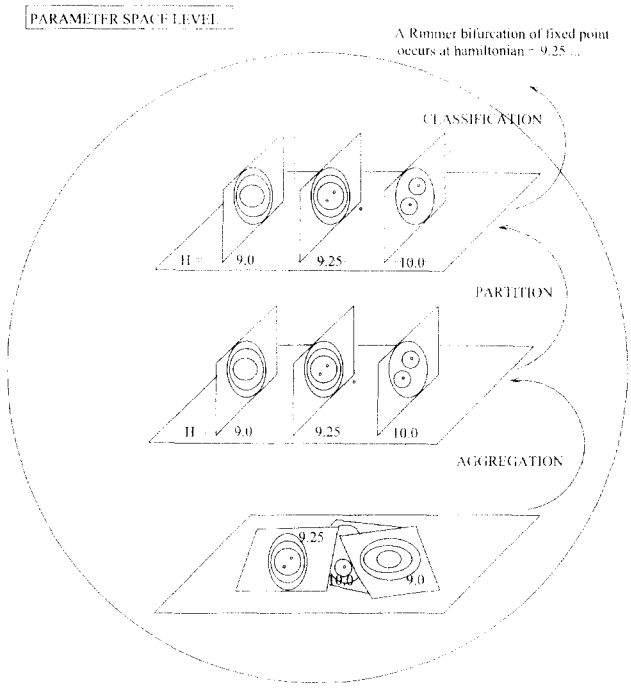


Fig. 8. The collection of phase portraits are ordered according to their parameter values. Topological changes between neighboring phase portraits are described symbolically. The symbolic description is matched against a list of known bifurcation patterns. The two phase portraits inside the dotted box are topologically equivalent.

topological change in the phase portrait, such as a change in the number or stability of fixed points. So the purpose of the parameter space level is to find a list of topologically different phase portraits such that the topological changes can be explained by some known type of bifurcation. Parameter space searching consists of three steps (Fig. 8):

(1) aggregating phase portraits;
(2) partitioning phase portraits into groups;
(3) classifying each group according to bifurcation patterns.

Aggregation. The first step of parameter space searching is the aggregation of phase portraits. Its purpose is to find out the adjacency relations among phase portraits. For a single-parameter mapping whose parameter is a real number, the aggregation is simple. This is because in a one-dimensional space we have a nice definition of adjacency: two phase portraits with parameter values p_1 and p_2 are neighbors in the parameter space if there is no other phase portrait with parameter value p_3 such that $p_3 \in (p_1, p_2)$. Such a relation can be determined algebraically without resorting to elaborate geometric algorithms like those used in orbit recognition or phase

space searching. Like the orbit consistency checking on the phase space level, there is a domain-specific pairwise phase portrait consistency checking. When two neighboring phase portraits are found to be inconsistent, a new parameter value will be proposed for searching out the missing phase portrait(s).

Partition. The second step is to describe the topological changes between two neighboring phase portraits symbolically. Phase portraits that are topologically equivalent are grouped into the same equivalence class.

Classification. The final step is the assignment of known bifurcation patterns to subsets of the entire collection of phase portraits. The classification is obtained by matching symbolic descriptions of topological changes against a list of bifurcation rules.

Upshot

This section is about the underlying computational processes that explain KAM's behavior. I have omitted a lot of details in the description; they are left for the remainder of this paper. The important point is that KAM has a simple computational structure. It is organized into three different semantic levels—the orbit level, the phase space level, and the parameter space level. Each level has the *same* pattern of behavior. Specifically, within each level, spatial properties and relationships that are not explicitly represented in the initial representation are extracted by applying three operations—(1) aggregation, (2) partition, and (3) classification—iteratively.

Two general observations can be made about these three operations. The first observation is that aggregation depends on domain-independent geometric knowledge, while classification is completely domain-specific. Partition has an intermediate character: it depends partly on domain-independent knowledge, and partly on domain-specific knowledge.

The second observation is about the relationship between geometric and symbolic representation. The type of geometric manipulations that can be done on an object depends on what is known about it semantically. For instance, if we know a given point set represents a quasiperiodic orbit, then we can compute properties of the orbit such as its curvature and the area enclosed by it. Such properties may not be meaningful for any arbitrary point set. Symbolic descriptions facilitate the process of classification, the assignment of semantic category, because matching symbolic descriptions is much simpler than matching geometric structures directly. So it seems that symbolic description is derived from a geometric representation of an object for the purpose of matching and classification. Once its semantic category is known, further shape and spatial information can be deduced about the object by more specific geometric operations.

4. KAM exploits vision techniques to classify orbits

The KAM program applies judgement similar to that of an expert dynamicist in directing the course of its numerical experiments. In making judicious choices of what to try next, KAM must interpret what it sees. The images of an initial point produced by iterating the map form a set of isolated points. This orbit must be classified. KAM must be able to extract shape and clustering information from the point set. It must determine whether the point set covers a closed curve, a cluster of closed curves, or a two-dimensional region. It must be able to estimate the centroid and area enclosed by a curve and to recognize the shape of a curve. It must aggregate the components of an orbit so that it can be further classified. It must also be able to determine the number of clusters in an orbit, as this number gives the period of the associated periodic orbit. KAM implements these abilities with techniques from computer vision and computational geometry.

4.1. Topological ideas simplify the ontology

Consider a system of many bodies moving under the influence of force law.. A fundamental idea in dynamical systems theory is a method for representing the state of such a system [4]. Imagine that the state of the system is represented by a point in *phase space*, a high-dimensional space whose axes are the positions and velocities of all the bodies. As the system evolves in time, it traces out a curve in phase space.[1] Mathematicians gives this curve a name—the *trajectory* corresponding to the initial state. Another name is *orbit*. The latter name is probably motivated by analogy with planet motions.

If the system's motion is periodic, the curve in phase space must close up in a loop. That is, the orbit must be a closed curve. By using the language of topology, we turn the analytical question about the existence of periodic solutions to differential equations into a question about the topological properties of an orbit.

But unless we have a way to find closed orbits in phase space, we won't be making too much progress in finding periodic solutions. And here is where the power of the topological viewpoint comes in. Let us first assume the phase space is three-dimensional for the sake of easy visualization. (The idea involved can easily be generalized to higher dimensions.) Imagine slicing the phase space with a plane. Let us follow the orbit starting from some point on this plane. If the motion is periodic, then the curve must return to the plane at its exact starting point. The curve may intersect the plane

[1] Some related works in qualitative physics on using phase space include Lee and Kuipers [7], Struss [17], and Sacks [15]. However, their techniques have only been applied to phase space of second-order continuous systems, which cannot exhibit any chaotic behavior.

several times before it hits its starting point, or it may do some wild things before it comes back. But the important point is we don't have to worry about these details. The question we really want to ask is: Does the curve come back to its exact starting point? If it does, we get a periodic solution.

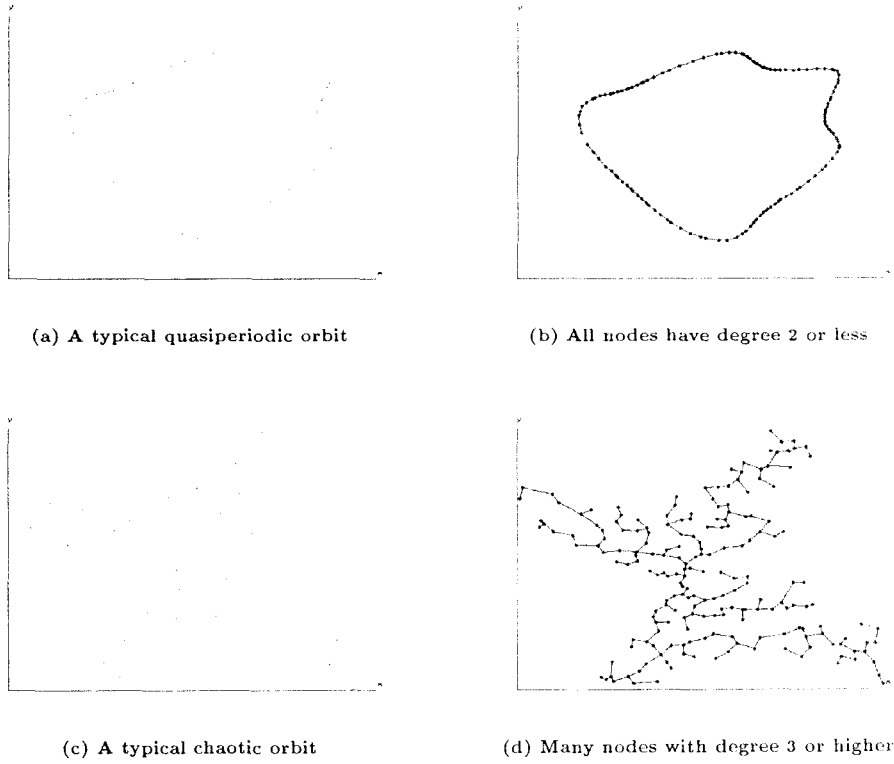
This plane is known as the *Poincaré section*, after its inventor. It has an obvious n-dimensional generalization: an $(n-1)$-dimensional hypersurface embedded inside an n-dimensional phase space.

The Poincaré section is a remarkable idea. The topological viewpoint allows us to get at a periodic solution without worrying about the detailed dynamics. Instead of following the entire orbit, we can just focus on how points on the section are mapped onto each other, ignoring pretty much everything else that happens between the time it leaves the section and the time it comes back. This mathematical trick buys a lot of simplification. For one thing, we don't have to deal with differential equations governing the motion in phase space, but rather *discrete* mappings defined on the Poincaré section, which has one fewer dimension. More important, mapping on a Poincaré section—Poincaré mapping, for short—in some sense capture the complete dynamics of the original system. That is to say the qualitative behavior of the continuous "flow" in phase space can be completely determined from its Poincaré mapping [1].

A fundamental property of a classical Hamiltonian system with two degrees of freedom—think of it as two coupled oscillators with no friction—is the area-preserving property: a bundle of initial points covering a small region is mapped onto another region with the same area. The shape of the region may change, but the area is invariant under the map. This property severely limits the possible long-time behaviors of orbits. In particular, there are only four possible types of orbits to distinguish. On a Poincaré section, periodic orbits appear as isolated points, quasiperiodic orbits appear as closed curves or island chains, homoclinic orbits (also known as separatrices) appear as curves connecting up the hyperbolic periodic points, and chaotic orbits appear to fill regions of two-dimensional space.

4.2. A quasiperiodic orbit can be distinguished from a chaotic orbit by the branching factors in the minimal spanning tree

KAM classifies orbits using methods based on the Euclidean minimal spanning tree (MST)—the tree that interconnects all the points with the minimal total edge length—which it constructs by the Prim–Dijkstra algorithm [2]. For each subtree of the spanning tree, KAM examines the degree of each of its nodes connected to it in the subtree. For a smooth curve, the spanning tree consists of two terminal nodes of degree one and other nodes of degree two. For a point set that fills an area, its corresponding spanning tree consists of many nodes having degree three or higher (Fig. 9).



(a) A typical quasiperiodic orbit

(b) All nodes have degree 2 or less

(c) A typical chaotic orbit

(d) Many nodes with degree 3 or higher

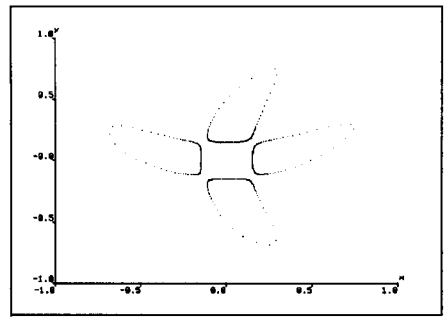
Fig. 9. Starting with successive iterates of a point, KAM classifies orbits using algorithms from computer vision. As shown above, a quasiperiodic orbit can be distinguished from a chaotic orbit by examining the branching factors in the Euclidean minimal spanning tree.

4.3. Components of a periodic orbit can be isolated by deleting excessively long MST edges

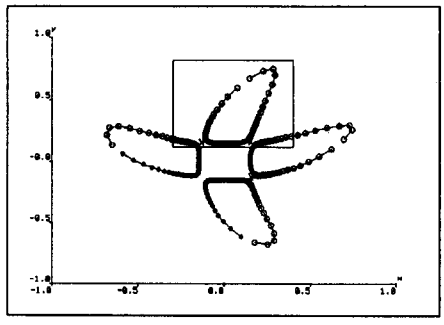
To aggregate points, KAM deletes the tree edges that are significantly longer than nearby edges, following an aggregation algorithm suggested by Zahn [24]. Such an excessively long edge is called an inconsistent edge. It can be detected by comparing its length with the average length of its nearby edges. Deleting the inconsistent edges results in several isolated connected components of the MST. Furthermore, if each of these isolated components resembles a quasiperiodic orbit, then KAM can classify the original tree as representing an island chain (Fig. 10).

4.4. Qualitative shape of a quasiperiodic orbit can be obtained by scale-space filtering techniques

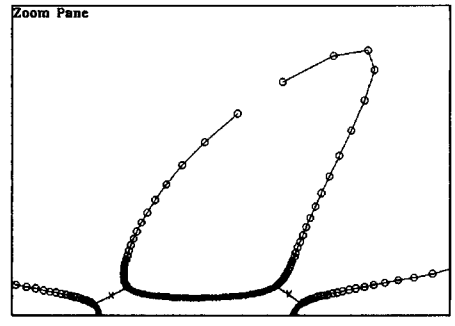
To compute the area and centroid of the region bounded by a curve, KAM generates an ordered sequence of points from the spanning tree, and spline-interpolates the sequence to obtain a smooth curve. Straightforward



(a) 512 iterates of an island chain



(b-1) MST: solid dots denote diameter nodes
circles denote non-diameter nodes



(b-2) Magnifying the boxed region
Crosses (x) are inconsistent edges.

```
Trying 512 iterates...
5 out of 6 possible near-diameters are tried.
Total nodes = 512.  Number of diameter nodes = 339.  Branching nodes = 6.
Branching-histogram = (0 ...  0 30 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
                       0 0 30 30 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 27 30 0 0 0 0 ...)
Diameter structure is (LINE BRANCH BRANCH LINE BRANCH BRANCH LINE BRANCH BRANCH LINE).
Percentage of nodes on main stem and primary branches = 100.0%.
Percentage of branch nodes on diameter = 1.7699127%.
Clusters = 4.  Inconsistent edges = 3.  Longest inconsistent edge = 0.06967917.
The orbit has periodic structure with period = 4.
```

(c) Symbolic description of MST

Fig. 10. KAM uses a clustering algorithm to group iterates into components. The components of an island chain can be isolated by detecting inconsistent edges from the MST and deleting them from the graph.

algorithms are then applied to compute the area or centroid. Shape recognition is accomplished using scale-space methods pioneered by Witkin [21]. For instance, to detect rapid changes in boundary curvature (Fig. 11), KAM uses the following method. The "curve" formed by the iterates of an orbit is parameterized by $C(s) = (x(s), y(s))$ where s is the arc length. The coor-

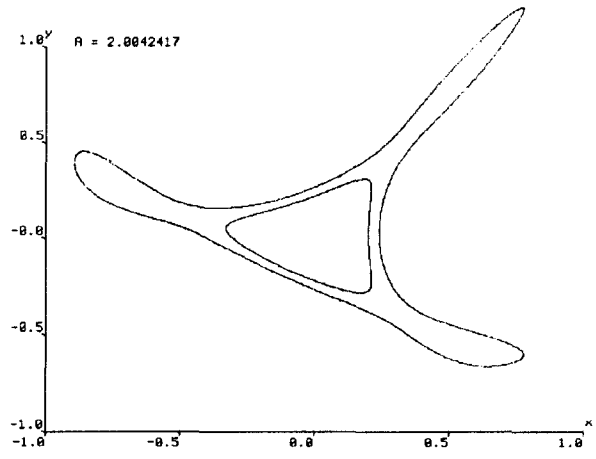


Fig. 11. KAM distinguishes the outer "starfish" orbit from the inner triangular orbit. The starfish orbit is characterized by its three long radially disposed arms about a central disk. The arms can be detected by rapid changes in the curvature of the orbit boundary. The length of an arm relative to the radius of the central disk provides a measure of the protrusiveness of the arm.

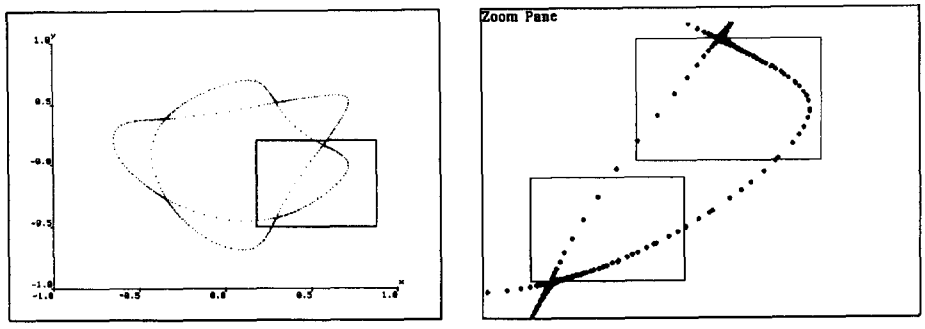
dinate functions, $x(s)$ and $y(s)$, are smoothed by the Gaussian and its first two derivatives at several spatial scales. Finally, the zero-crossings of the curvature function $\kappa(s)$, and the signs of $\dot{\kappa}(s)$ are computed to determine the locations and type of the extrema.

4.5. A separatrix can be identified by the geometry of its branching structure

A separatrix is sort of intermediate between a smooth one-dimensional curve and a two-dimensional region: its MST has branching nodes, but these branching nodes have distinctive structures. In particular, along a diameter of the MST representing a separatrix, there are visually prominent (long) branches. Moreover, these branches come in pairs (see Fig. 12).

To capture the branching structure of a diameter, it is useful to compute the *branching histogram*, which gives, for each node on the diameter, the length (in terms of the number of edges) of the longest branch from that node. For example, the branching histogram for a MST that has no branching node is just a whole bunch of zeros. In general, a long run of zeros in the branching histogram suggests a line structure, while peaks in the histogram indicates long branching from the diameter. Small numbers in the histogram, such as 1 or 2, mean the presence of short branches.

Instead of looking at a branching histogram, which is really a long sequence of numbers, it is more useful to extract qualitative information from it and represent symbolically the branching structure of the MST. The language used to describe the branching structure turns out to be quite simple; it consists of all finite sequences of three symbols—LINE, SHORT-LINE, and



(a) A separatrix with 5 loops; one of the loops is inside the boxed region

(b) Combining two branches from adjacent hyperbolic segments to form a loop

Fig. 12. Branching structure of a separatrix. Notice that the points on the two sides of a neck have a distinctive geometry: they lie on the two branches of a hyperbola. Let me call this arrangement a hyperbolic segment. The five-loop separatrix can then be thought of as consisting of five such hyperbolic segments.

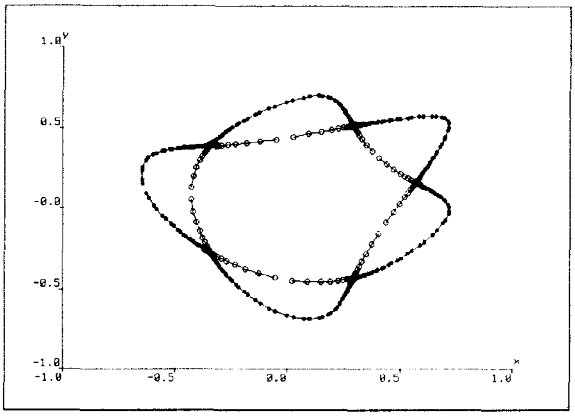
BRANCH. Informally, a LINE denotes a segment of the diameter such that none of the nodes in the segment has a branching depth greater than 2. A SHORT-LINE is a LINE such that the path length of the segment is smaller than a certain threshold. A BRANCH denotes a singleton diameter node whose branching depth is greater than 2.

Let us look at an example. Fig. 13(a) shows the MST representation of a separatrix with 640 iterates. The diameter of the MST consists of 438 nodes which are denoted by the large solid dots. The branching histogram—a sequence of 640 numbers—is shown in Fig. 13(b). The branching structure is represented symbolically by a compact sequence of symbols consisting of 21 symbols. Note that there are 10 BRANCHes in the symbolic description and that these branches come in pairs in the sense that there is a short line between two branches in a pair, while the pairs themselves are well separated from each other.

Let's define more precisely what a hyperbolic structure is. A *hyperbolic segment* is a sequence of four symbols: "LINE BRANCH SHORT-LINE BRANCH". A *hyperbolic structure* is a finite sequence of hyperbolic segments.

HYPERBOLIC-SEGMENT
  = "LINE BRANCH SHORT-LINE BRANCH",
HYPERBOLIC-STRUCTURE
  = { HYPERBOLIC-SEGMENT}*.

Finally, any MST whose branching structure is a hyperbolic structure, possibly with an optional LINE symbol in the end, is identified as the representation of a separatrix. The number of loops of the separatrix is the same as the number of hyperbolic segments in the branching structure.



(a) The MST of 640 iterates: • = diameter nodes; o = non-diameter nodes

```
Trying 640 iterates...
Total nodes = 640.  Number of diameter nodes = 438.  Branching nodes = 44.
Branching-histogram =
(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 16 0 0 0 0 0 1 0 0 1 15 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 17 0 1 0 2 14 0 2 0 0 2 0 0 0 1 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16 1 0 0 1 16 0 0 2
0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 16 0 0 0 0 0
1 0 0 16 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0
0 0 0 0 16 0 0 0 0 1 0 0 0 15 1 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0)
Diameter structure is ( LINE BRANCH SHORT-LINE BRANCH LINE BRANCH SHORT-LINE BRANCH
                        LINE BRANCH SHORT-LINE BRANCH LINE BRANCH SHORT-LINE BRANCH
                        LINE BRANCH SHORT-LINE BRANCH LINE).
Percentage of nodes on main stem and primary branches = 98.59375%.
Percentage of branch nodes on diameter = 9.589043%.
Clusters = 5.  Inconsistent edges = 0.  Loops = 5.
```

(b) Branching histogram and symbolic description of the MST

Fig. 13. A symbolic description of the MST of a 640-iterate separatrix is derived from the branching histogram. The diameter of a separatrix has a hyperbolic structure. Note that a large number (such as 14, 15, etc. in this example) in the histogram indicates a long branch.

4.6. Stable symbolic descriptions of the MST can give a reliable classification

KAM has to decide when it has gathered enough iterates to reliably classify an orbit. There is a tradeoff involved. In the limit of using a very large number of iterates, a reliable classification is achieved because the probability of missing part of the orbit structure goes to zero. On the other hand, a classifying procedure operating on a large number of iterates tends to

be prohibitively expensive. So, a practical procedure should use the smallest number of iterates consistent with certain specified reliability criterion.

With the MST representation of the iterates, a useful reliability criterion is the stability of the symbolic description of the branching structure of the MST. For a given set of iterates, we compute the branching description, i.e., a list of symbolic descriptors (such as LINE, SHORT-LINE, or BRANCH) that describe the branching structure of the diameter (Fig. 10(c)) . Then we add more iterates to the set, compute a new branching description, and check if the two branching descriptions are the same. If the branching description remains invariant, we assume steady state is reached. Otherwise, we progressively enlarge the set of iterates until there is no further change in the branching structure.

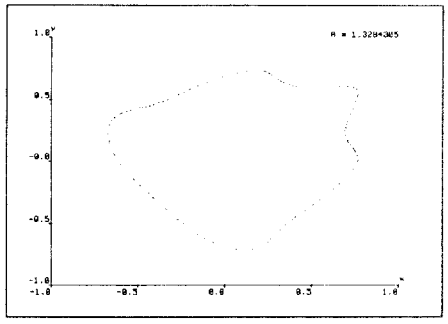
For the thousands of orbits that have been tried, the recognition algorithm based on MST never gives a wrong answer. The algorithm may give up and fail to give an answer, but apparently it never assigns a wrong orbit category. Figure 14 shows two examples to illustrate the robustness of the algorithm. In Fig. 14(a-1), the set of 256 iterates is identified as a quasiperiodic orbit. Figure 14(a-2) below displays 50,000 iterates starting at the same initial condition. We notice that the overall shape of the orbit is unchanged with the exception that the spacing between neighboring iterates becomes so small that it gives an impression of a solid closed curve.

The second example is a set of 800 iterates in Fig. 14(b-1). The recognition algorithm identifies this set as a five-loop separatrix. Again, the overall shape of the orbit does not change much after 50,000 iterates as shown in Fig. 14(b-2).

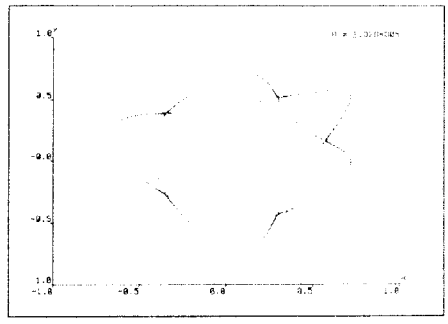
5. KAM exploits deep mathematical knowledge to constrain search in phase space

Dynamical behavior is complicated, but it is not arbitrary. There is structure in the phase space that restricts the ways in which trajectories can be pieced together to form a legal phase portrait. In Hamiltonian systems the evolution of the phase space is area-preserving, i.e., on a surface of section, any region of the phase space will be transformed to another region with the same area. The area-preserving property has important dynamical consequences. It is this kind of knowledge that enables dynamicists to infer a good understanding of the phase portrait from only a small, but well chosen set of experiments.

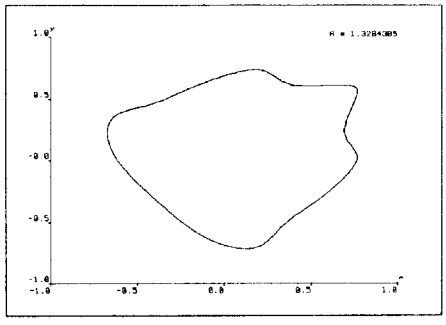
The phase portrait in Fig. 15, taken from a historically important paper in dynamics by Henon [5], describes how adding a simple quadratic nonlinearity to a linear rotation can lead to dramatic changes in dynamical behavior. The figure characterizes the dynamics by showing only a few



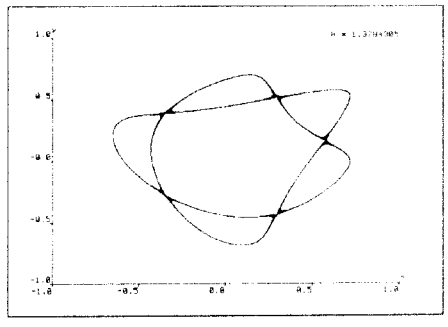
(a-1) KAM identifies these 256 iterates as a quasiperiodic orbit.



(b-1) KAM identifies these 800 iterates as a separatrix.



(a-2) Shape is unchanged after 50000 iterates.



(b-2) Shape is unchanged after 50000 iterates.

Fig. 14. Showing the robustness of KAM's recognition result. Both the quasiperiodic orbit (a) and the separatrix (b) retain their shape after 50,000 iterates.

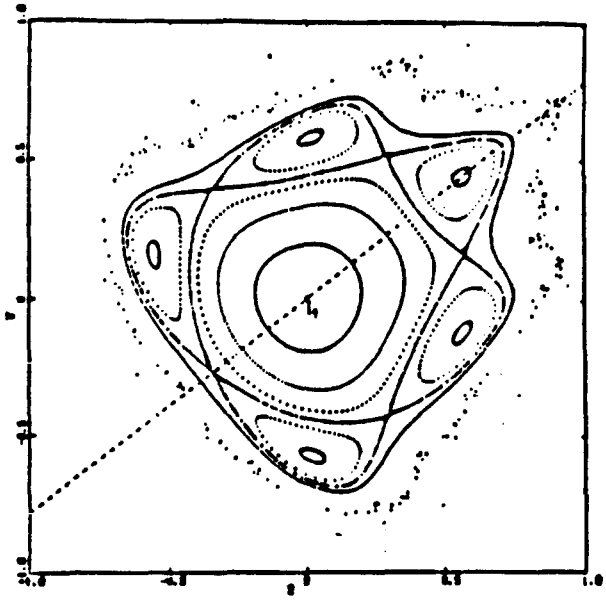
orbits. Presumably, Henon was able to generate this figure after only a few judiciously chosen numerical experiments.

The KAM program analyzes systems in the same way [22]. It knows enough about the constraints on the structure of phase space to choose initial conditions and parameters as cleverly as an expert dynamicist. KAM's summary description of Henon's map, shown in the upper panel of Fig. 16, is almost identical to the summary presented by Henon. Moreover, KAM was able to deduce this description after trying only ten initial conditions.

5.1. Local inconsistency of trajectories can be used to deduce missing structures

Figure 17(a) depicts a flow pattern containing two concentric flow lines. The two flow lines circulate in opposite directions: the outer one (A) goes clockwise; the inner one (B), counterclockwise. The flow pattern, as it stands, is not consistent: the flow lines are rubbing against each other. Some important features must be missing. Let me elaborate.

We first draw a vertical line, L, through the two circular flow lines. At



HENON MAP

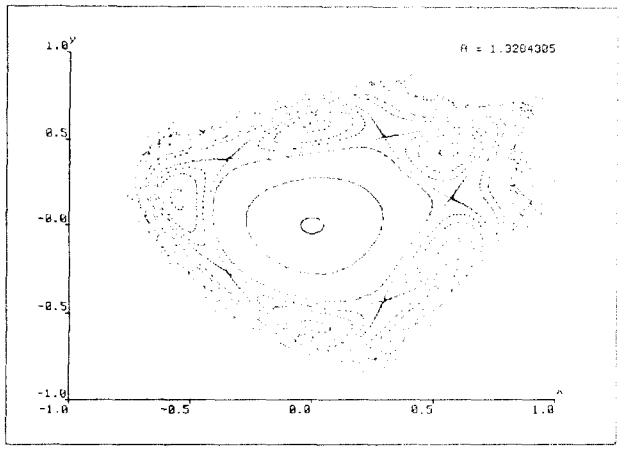
$$x_{n+1} = x_n \cos\alpha - (y_n - x_n^2)\sin\alpha$$
$$y_{n+1} = x_n \sin\alpha + (y_n - x_n^2)\cos\alpha$$

where $x_n \in (-1\ 1), y_n \in (-1\ 1)$ and $\alpha \in (0\ 2.2)$.

Fig. 15. The phase portrait for Henon map with $\alpha = 1.3284305$. This picture is directly taken from Henon's paper. Compare this with the one that KAM finds.

point p where L intersects flow line A, the tangential component of the velocity points to the right. However, at point q where L intersects flow line B, the tangential velocity component points left. By continuity of velocity, the tangential velocity component must vanish at some point on L between p and q. Call this point c. What about the radial velocity at c?

There are two possibilities. First, the radial component of velocity at c is also zero. In this case, we have a *stagnant point.* Second, and more likely, the velocity has some nonzero radial component. Let us suppose the velocity vector goes outward (Fig. 17(b)). Because the fluid is incompressible, the outgoing fluid particles cannot accumulate or disappear; they must change direction and turn inwards at some point. We therefore infer that there must be a roller of some sort that allows fluid particles to go clockwise half of the time, and counterclockwise the other half (see Fig. 17(c)). Continuously shrinking the roller, we see that a stagnant point must exist inside the roller. This type of stagnant point is called *elliptic* because it is surrounded by



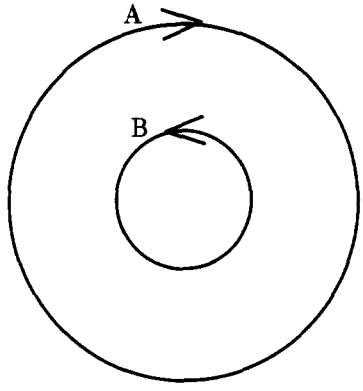

```
The phase portrait has 1 family of bifurcating orbits.
***
FIRST BIFURCATING FAMILY:

The portrait has an elliptic fixed point at (0.000 0.000).
Surrounding the fixed point is a regular region bounded by a KAM curve with
rotation number between 1/5 and 1/4.
Outside the regular region lies a chain of 5 islands.
The island chain is bounded by a KAM curve with rotation number between 4/21 and 5/26.
The outermost region is occupied by ESCAPE orbits.
Chaotic orbits occupy 62% of phase space.
```

Fig. 16. Upper: KAM produces the phase portrait for for the Henon map with $\alpha = 1.3284305$ (α is shown as "A" in the figure). Lower: KAM generates a summary description of the phase portrait.

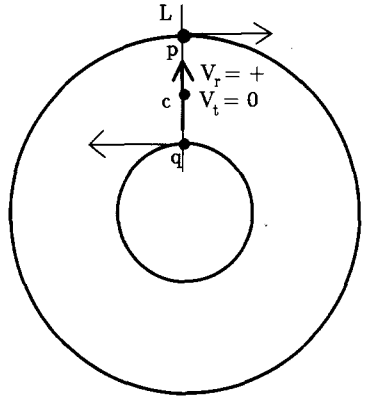
circular or elliptical flow lines.

Next, we draw a second vertical line L' crossing flow lines A and B at point p' and q' respectively. We repeat the same argument to deduce that another roller and stagnant point must exist (see Fig. 17(d)). Is the flow pattern augmented with the rollers consistent so far? No. Consider what happens when the two rollers get really close to each other. Where they touch, the velocity components cancel one another; a third stagnant point is formed (Fig. 17(e)). Moreover, this last stagnant point is called *hyperbolic* because it is the limit of a *separatrix-like* flow line enclosing the rollers. Continuing to add rollers inside the annular region, we will eventually come back to the first roller. We see that in such a configuration the elliptic and hyperbolic stagnant points must alternate and come in pairs (Fig. 17(f) shows six pairs of such stagnant points).

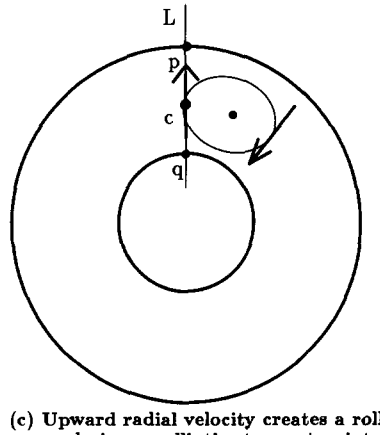
Inserting the "missing" features—the rollers, alternating elliptic and hyperbolic stagnant points, and the separatrix—into the original flow pattern, we obtain a globally consistent picture (see Fig. 17). Notice that we have arrived



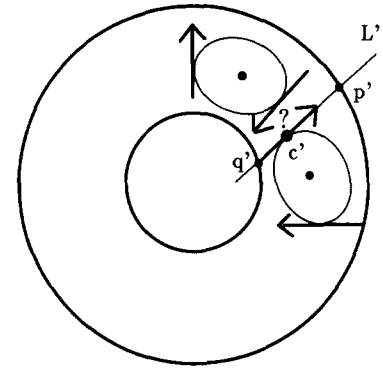
(a) Two concentric flow lines circulating in opposite direction



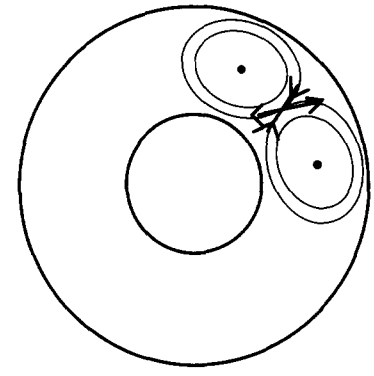
(b) Tangential velocity vanishes at c. Assume radial velocity is upwards.



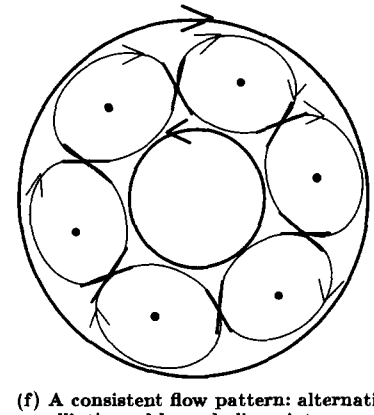
(c) Upward radial velocity creates a roller enclosing an elliptic stagnant point



(d) Feature missing between two rollers



(e) Separatrix-like flow lines meeting at a hyperbolic point



(f) A consistent flow pattern: alternating elliptic and hyperbolic points

Fig. 17. From the observation that two neighboring orbits are incompatible (as shown in (a)), KAM deduces not only that there must be missing structures in the region between them, but also that the structures must be in the form of island chains with alternating elliptic and hyperbolic periodic points (as shown in (f)). This is essentially the content of the Poincaré–Birkhoff theorem.

at this nontrivial conclusion by arguing about the physical continuity of adjacent flow lines and their incompressibility nature—an example of the powerful idea that global conclusions can be deduced from strictly local decisions.

5.2. *Structure theorems are incorporated in grammatical rules*

The inconsistency arguments just presented depend on a continuity condition on neighboring trajectories. However, a trajectory on a section of section is not a smooth line, but consists of points jumping from one place to another. To formulate an analogous continuity condition for the discrete flow, KAM uses sophisticated mathematical invariants to incorporate structure theorems such as the Poincaré–Birkhoff theorem [1].

One such invariant is the *rotation number* of an orbit, a quantity that measures the asymptotic average of the angular distances between any two successive iterates in units of 2π radians. Simple grammatical rules based on rotation number can be written to capture the structure theorems. Here are some examples of these grammatical rules (see Fig. 18):

Rule 1 (*Missing Islands*). Two quasiperiodic orbits with incompatible rotation numbers are not consistent.

Rule 2 (*Missing Separatrix*). An island chain and a quasiperiodic orbit are not consistent: a separatrix is missing.

Rule 3 (*Missing Quasiperiodic*). Two island chains with different number of islands are not consistent.

Rule 4 (*Missing Orbit-1*). A separatrix with k loops is inconsistent with a quasiperiodic orbit having rotation number ρ unless k is compatible with ρ.

Rule 5 (*Missing Orbit-2*). A separatrix with k loops is inconsistent with an island chain with m islands unless $k = m$.

Rule 6 (*Missing Orbit-3*). Two separatrices having different number of loops are not consistent.

Rule 1, for instance, embodies a theorem that the rotation number of nearby orbits must change continuously [9]. As an example of how it is used, suppose that KAM has located two nearby quasiperiodic orbits having rotation numbers ρ_1 and ρ_2 respectively. Suppose ρ_1 is slightly smaller than $1/5$, and ρ_2 slightly larger. With only these two orbits, KAM's evolving phase picture cannot be complete. By continuity, there must exist a third, nearby orbit with rotation number exactly equal to $1/5$, i.e., a periodic orbit of period 5, which KAM proceeds to search for and classify.

5.3. *The grammar is implemented by Waltz's type of constraint analysis*

When KAM detects a pair of inconsistent neighboring orbits, it flags a complaint. A *consistency complaint* is a data structure describing the nature of the complaint. Specifically, it records the type of complaint, and the

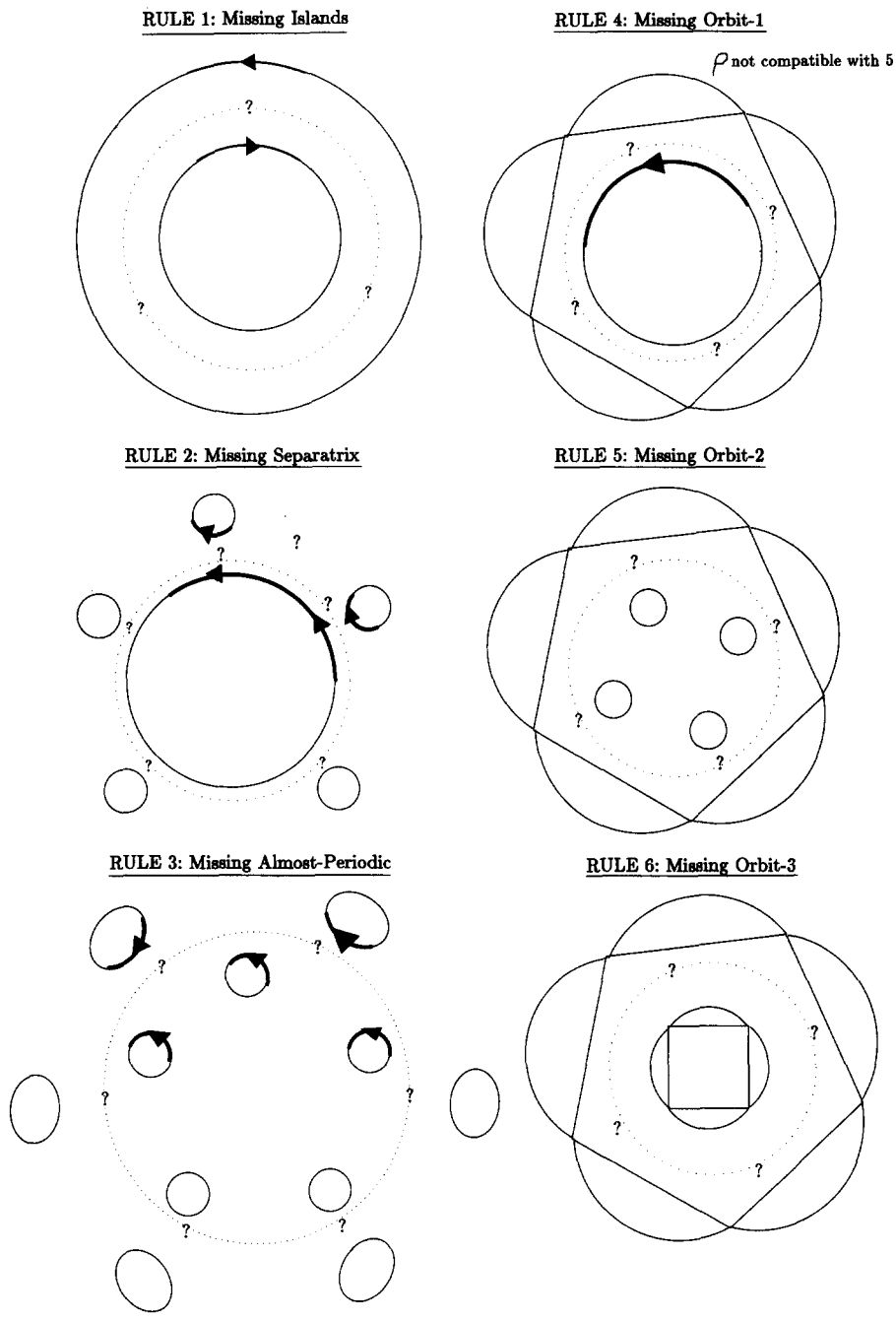


Fig. 18. KAM encodes structure theorems in six pairwise inconsistency rules. These rules enable KAM to detect inconsistency in the phase portrait and severely constrain the search for missing structures in the phase space.

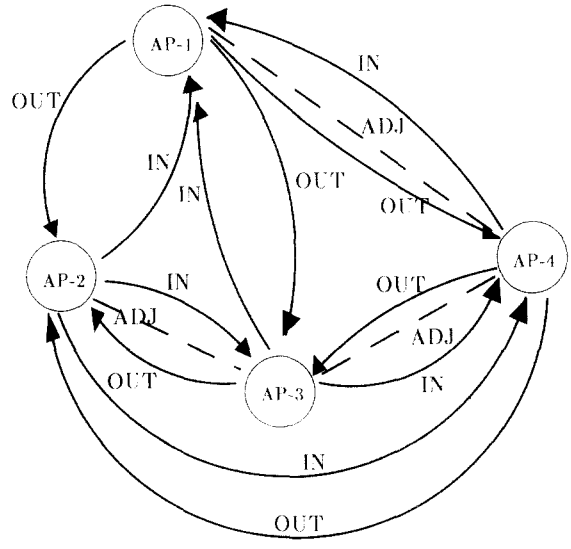


Fig. 19. A snapshot showing the internal representation of an evolving phase portrait. KAM represents the orbits and their spatial relationships by a network, the orbit adjacency graph. The nodes and links of the graph are updated whenever a new orbit is added to the phase portrait. The cross (×) on the left picture indicates the next initial state to be tried.

identity of the orbits involved. A *complaint-dispatcher* examines the type of a complaint, and invokes the appropriate suggestion rule to propose new candidate initial states.

The basic data structure is an *orbit adjacency graph* (Fig. 19). The graph has one type of node, and three types of links. Each node of the graph represents an orbit. The first two types of link—INSIDE and OUTSIDE—are directed. An INSIDE link goes from orbit A to orbit B if A is inside the region enclosed by B. A similar definition goes with the OUTSIDE link.[2] The third type of link is the ADJACENCY link. We say an adjacency link between two nodes is *valid* if the orbits in question are adjacent in the phase portrait. An adjacency link is *inconsistent* if the adjacency cannot be part of a legal flow pattern allowed by local dynamics.

Consistency maintenance is the process of updating adjacency links: create new links, remove invalid links, and identify inconsistent links. The process has two purposes:

(1) maintain correct adjacency relations between orbits as new orbits are added, and
(2) create a complaint for each inconsistent link.

The complaints are stored in a stack, the *complaint-stack*. A complaint is

[2]Note both INSIDE and OUTSIDE links are needed because they are not exactly inverses of each other. For example, two island chains can both be outside of each other.

removed from the stack if either the adjacency link causing the complaint is no longer valid, or the complaint does not lead to useful new orbits. KAM continually searches for new orbits until the complaint-stack is emptied.

The principal steps of consistency maintenance are:

Step 1. Initially, pick some random initial states. Create orbits corresponding to these states.

Step 2. Add the newly created orbits to the orbit adjacency graph.

Step 3. Update adjacency links. Produce a list of invalid adjacency links to be removed, and a list of new adjacency links to be added.

Step 4. For each new adjacency link to be added, run inconsistency rules against it. If the link is inconsistent, a new complaint is made. Put the complaint on top of the complaint-stack. Add the new link to the graph.

Step 5. For each invalid adjacency link to be removed, delete, if any, its associated complaint in the complaint-stack. Remove the link from the graph.

Step 6. Handle the complaint on top of the complaint-stack. Examine the type of complaint, and propose a list of new initial states to try.

Step 7. Create a new goal with the suggested initial states. Make new orbits starting from these initial states.

- If the new goal leads to no useful orbits [3], the goal is abandoned, and the associated complaint removed.

Step 8. Repeat the process (Steps 2 to 7) until the complaint-stack is empty.

The consistency maintenance algorithm terminates when all the inconsistency complaints are removed. However, a consistent orbit adjacency graph—a graph all of whose orbits are pairwise and singly consistent—is a necessary but not sufficient condition for the self-consistency of a phase portrait. For instance, KAM will ignore an inconsistency complaint if it does not find any useful orbits in a few consecutive trials (see Step 7 above). One common situation that causes KAM to abandon a complaint is when KAM is looking for a "hard" separatrix—a separatrix that takes over a few thousand points to reveal its structure. Another common situation is when the region between two offending orbits is so small that the new orbit found is equivalent to one of the orbits already seen.

5.4. Orbit adjacency can be determined by contour activation

Roughly speaking, two orbits are adjacent if there is no orbit between them. KAM implements this idea of adjacency by a contour activation procedure. A phase space is internally represented as a rectangular grid. To

[3] An orbit is useless if either it is equivalent to some orbit previously found, or KAM fails to recognize its type.

find the neighboring orbits of a given orbit A, the grid cells surrounding orbit A are activated. The activation spreads both outwards and inwards, iteratively activating neighboring grid cells (see Fig. 20). The outward (inward) activation is terminated when another orbit outside (inside) A is encountered.

Because orbits are not really a continuous line (they are a collection of discrete points), it is possible for the activation to leak through a quasiperiodic orbit, and activate grid cells outside the orbit. Some sort of thresholding operation is therefore necessary to reduce such leakage. In the current implementation, the contour activation allows activation to continue until a sufficient number of the activated grid cells are "blocked" by the orbits encountered.

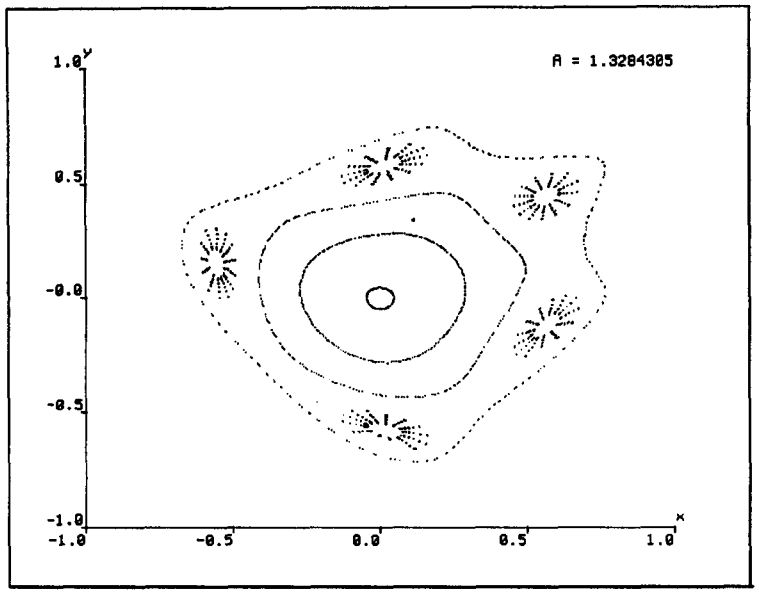
5.5. KAM will miss small structures in the phase portrait

The phase portrait for an area-preserving map is considerably more complicated than that apparent from Fig. 3. First of all, the Poincaré–Birkhoff theorem [8] tells us that typically there are alternating chains of elliptic and hyperbolic periodic orbits corresponding to *every* rational rotational number. Secondly, the island structures of an area-preserving map are *self-similar*: the geometric structure of islands surrounding fixed point or periodic point at one length scale is repeated at another length scale. This hierarchy of island structure exists in principle down to the smallest length scale [23, Chapter 4].

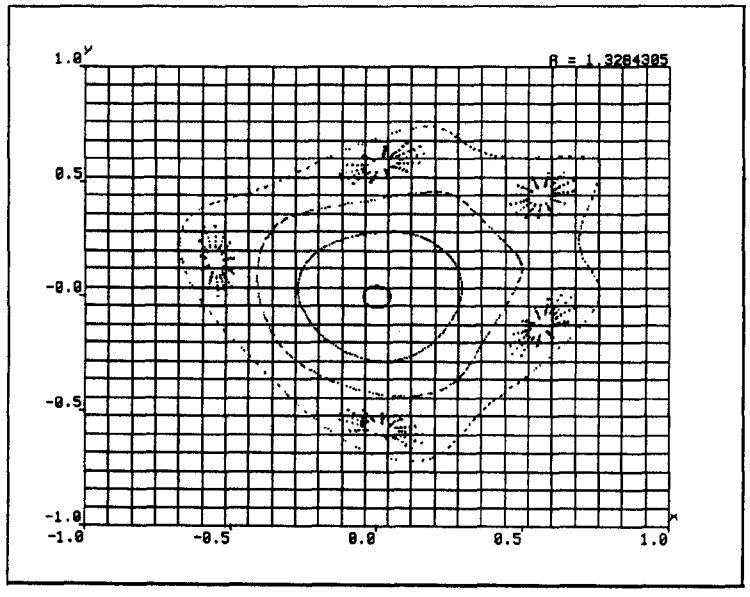
The complexity of island structures precludes any attempt to produce a *complete* phase portrait, i.e., a phase portrait containing representative orbits for every possible island chain or periodic orbit. Fortunately, for the purpose of a gross qualitative understanding of the phase space dynamics, the details of the island structures do not matter because these higher-order structures occupy small phase space area. The kind of qualitative description generated by KAM is sufficient for characterizing the gross behavior of the dynamical system under most (in the measure-theoretical sense) initial conditions.

6. KAM draws upon knowledge of typical changes of phase portraits to explain and predict bifurcation behavior

Dynamicists represent the qualitative changes in behavior by topological changes in the phase portrait. For example, changing a system parameter might cause an island chain to appear in a phase portrait, which signifies the birth of a stable periodic orbit. Each time when the topology of the phase portrait changes, we say a *bifurcation* occurs. For a parametrized family of dynamical systems, tracing the changes in steady-state orbits (such as fixed points and periodic orbits) provides a valuable summary of the family's behavior.



(a) Propagating the contour of the 5-island chain outwards



(b) Showing the underlying 25 x 25 grid.

Fig. 20. KAM decides whether two orbits are adjacent by a geometric algorithm that propagates the contour of an orbit. For example, to find what orbits are adjacent to the 5-island chain in (a), the contour is spread both outwards and inwards. Orbits first encountered by the activation are considered neighbors of the island chain.

Much research in nonlinear dynamics is devoted to studying such bifurcations. For one-parameter families of area-preserving mappings, the generically encountered bifurcations have been classified and are well understood. Knowledge of generic bifurcations enables a dynamicist to explore the parameter space efficiently because not only he knows what bifurcations to expect, but also he can decide which sequences of topological changes in phase portraits are legal. In a similar manner, KAM draws upon knowledge of typical changes to constrain its search in the parameter space.

6.1. *A small library of geometry of changes in flow pattern explains the commonly observed bifurcations*

Generic bifurcations of steady-state orbits of a dynamical system are classified according to the way in which the multipliers of the period map or Poincaré map cross the unit circle. For one-parameter families of area-preserving maps, generic bifurcations come in five types [11]:

- *extremal bifurcation*, at which a pair of fixed points can disappear or vanish;
- *period doubling*, at which the period of an orbit doubles;
- *phantom-3-kiss*, at which a fixed point loses its stability by colliding with an unstable period-3 orbit;
- *phantom-4-kiss*, at which a fixed point loses its stability by colliding with an unstable period-4 orbit; and
- *Poincaré–Birkhoff bifurcation*, at which a pair of period-n ($n \geqslant 5$) orbits can appear or vanish (see Fig. 21).

As an example of how knowledge of generic bifurcations is used, suppose that KAM has found two phase portraits having a "starfish-like" orbit around a fixed point and a upside down triangular orbit around a fixed point respectively (as for example in the first and last flow pattern in the phantom-3-kiss bifurcation sequence (Fig. 21). KAM's evolving parameter space picture cannot be complete. By the bifurcation theorem, there must be three topologically different phase portraits between the original two, which KAM proceeds to find and classify. The global picture of behavior of the system is then obtained by piecing together theoretically acceptable sequences of bifurcations.

6.2. *Concise symbolic descriptions of flow patterns are key to simple matching of phase portraits*

To apply the bifurcation theorems, KAM has to classify a phase portrait and match it against a flow pattern in one of the bifurcation sequences. Matching a phase portrait directly with the flow pattern can be difficult because the phase portrait typically contains many redundant features. For

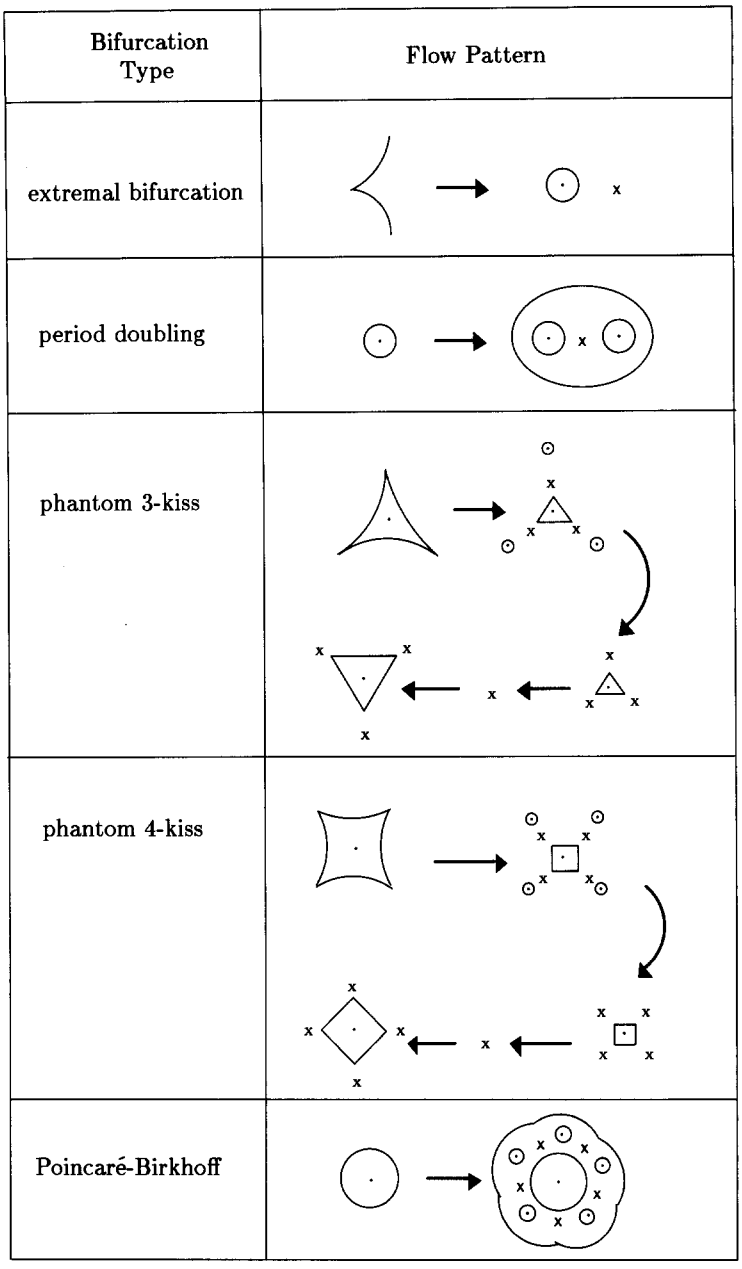


Fig. 21. Bifurcations of fixed points or periodic orbits of a dynamical system are classified according to the manner in which the multipliers of the mapping cross the unit circle in the complex plane. For one-parameter area-preserving maps, there are five generically encountered bifurcations. Dots (•) indicate elliptic (stable) fixed points. Circles with a dot inside (⊙) stand for islands. Crosses (×) are hyperbolic (unstable) fixed points.

example, if the phase portrait has a nested sequence of quasiperiodic orbits with no other types of orbit in between, there is no need to keep track of any quasiperiodic orbits other than the innermost and the outermost ones, i.e., the maximal ones.

A canonical flow pattern is a symbolic data structure for representing the essential qualitative features of a phase portrait. Orbits that are redundant—such as the nonmaximal island chains, and the nonmaximal quasiperiodic orbits—are removed to simplify the description. A canonical flow pattern is a group of essential orbits surrounding some primary orbit. It contains the following descriptions:

- the nature of the primary orbit (fixed point or periodic orbit);
- two indices m and n which indicate the number of island chains it has, and the number of those island chains that are bounded by some quasiperiodic orbit;
- the periods of the island chains;
- the rotation numbers and orientation of the maximal quasiperiodic orbits.

For an example, consider the flow pattern in Fig. 22. It has two island chains, a 5-island chain and a 6-island chain, surrounding a fixed point. The outer island chain is not bounded by any closed curve. So the flow pattern is classified as a FLOW-PATTERN-2-1. The *period* of a flow pattern is the list of the periods of the island chains that the flow pattern has. In this example, the period is (5 6). It has two maximal closed orbits. The inner one has rotation number in $(1/5, 1/4)$; the outer one in $(1/6, 1/5)$. So the rotation number of the flow pattern is $((1/5, 1/4), (1/6, 1/5))$.

6.3. *Efficient parameter space search can also be implemented by Waltz's constraint analysis*

The parameter space search algorithm is formally very similar to the phase space search algorithm. In either case, KAM computes the adjacency relations among some basic objects. Each of the adjacency relations is checked against a list of inconsistency rules (Fig. 23). If an adjacency relation is inconsistent, then the system looks for missing objects inside the region between the inconsistent objects to restore compatibility among adjacent objects. KAM continually searches for new objects until all the objects are pairwise consistent.

As an example of parameter space search, suppose that KAM has already found two phase portraits (Fig. 24) with parameter values equal to 1.996 and 2.173 respectively. With just these two phase portraits, KAM's evolving parameter space picture cannot be consistent. By the bifurcation theorem for phantom-3-kiss, there must be missing phase portraits in the interval

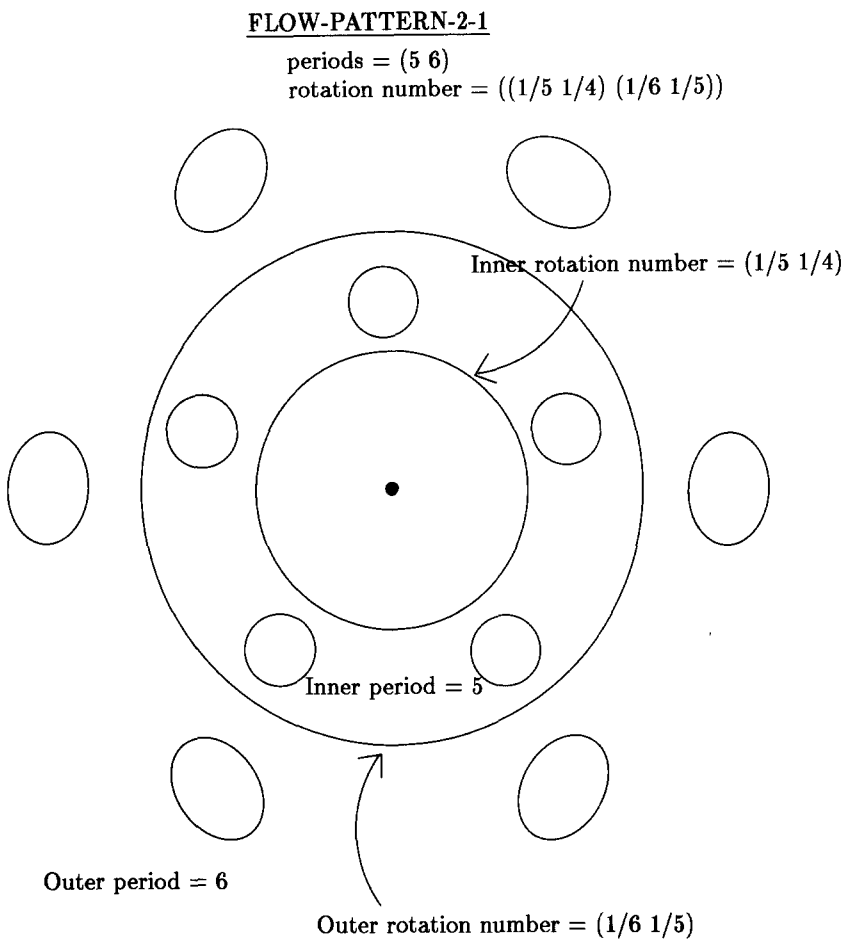
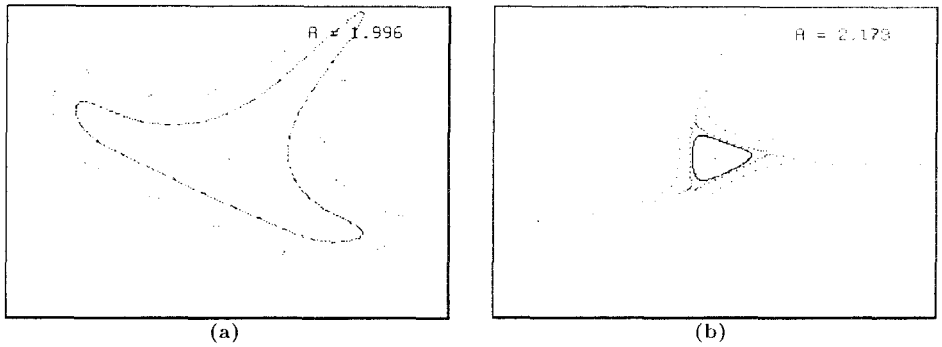


Fig. 22. KAM represents the important topological features of a phase portrait by a concise symbolic description consisting of the nature of the primary orbit, the periods of the island chains, and the rotation numbers and orientation of the quasiperiodic orbits.

RULE 004 **Period-doubling**
If (1) A is a F-FLOW-PATTERN
(2) B is a P-FLOW-PATTERN
(3) the period of the primary orbit of B is not equal to 2
then they are not consistent

RULE 005 **Phantom-3-kiss**
If (1) both A and B are phantom-3-kiss patterns
(2) A is flow-pattern-0-0
(3) B is flow-pattern-0-0
(4) A and B have very different orientations
then they are not consistent

Fig. 23. Two examples of grammatical rules that incorporate bifurcation theorems. Notice that a concise symbolic description of a flow pattern allows the grammatical rules to be expressed in a very simple manner. The grammatical rules are used in a constraint analysis similar to that used in phase space searching.



(a) (b)

Fig. 24. KAM recognizes that the two phase portraits cannot be consistent because they satisfy Rule 005, a rule that defines the inconsistency conditions for phantom-3-kiss bifurcation.

(1.996, 2.173). Therefore, KAM proceeds to search for and classify the missing portraits. Eventually, KAM finds a self-consistent sequence of eight phase portraits (Fig. 25).

7. Experiments

KAM has been tested on thousands of orbits and over thirty different phase portraits of the Henon map. It is able to reproduce completely all the phase portraits found in Henon's paper. Other experiments include:

(1) the standard map, whose phase space is a torus instead of the usual Euclidean plane,
(2) the Henon–Heiles system, a set of four differential equations that models the motion of a star in an axisymmetric galactic potential, and
(3) a wave tank problem.

(See [23] for details.)

A word about the running time for these experiments. It takes about one minute run time on a Symbolics 3640 to recognize a 256-iterate orbit, and about eight minutes for a 800-iterate separatrix. A typical phase portrait has about a dozen orbits, requiring approximately two to four hours of runtime.[4] A complete parameter space search can consume up to two days on the Symbolics.

[4]On the average, KAM takes about ten times longer to produce one phase portrait than a domain expert does. Approximately 80% of the running time, however, is spent on orbit recognition.

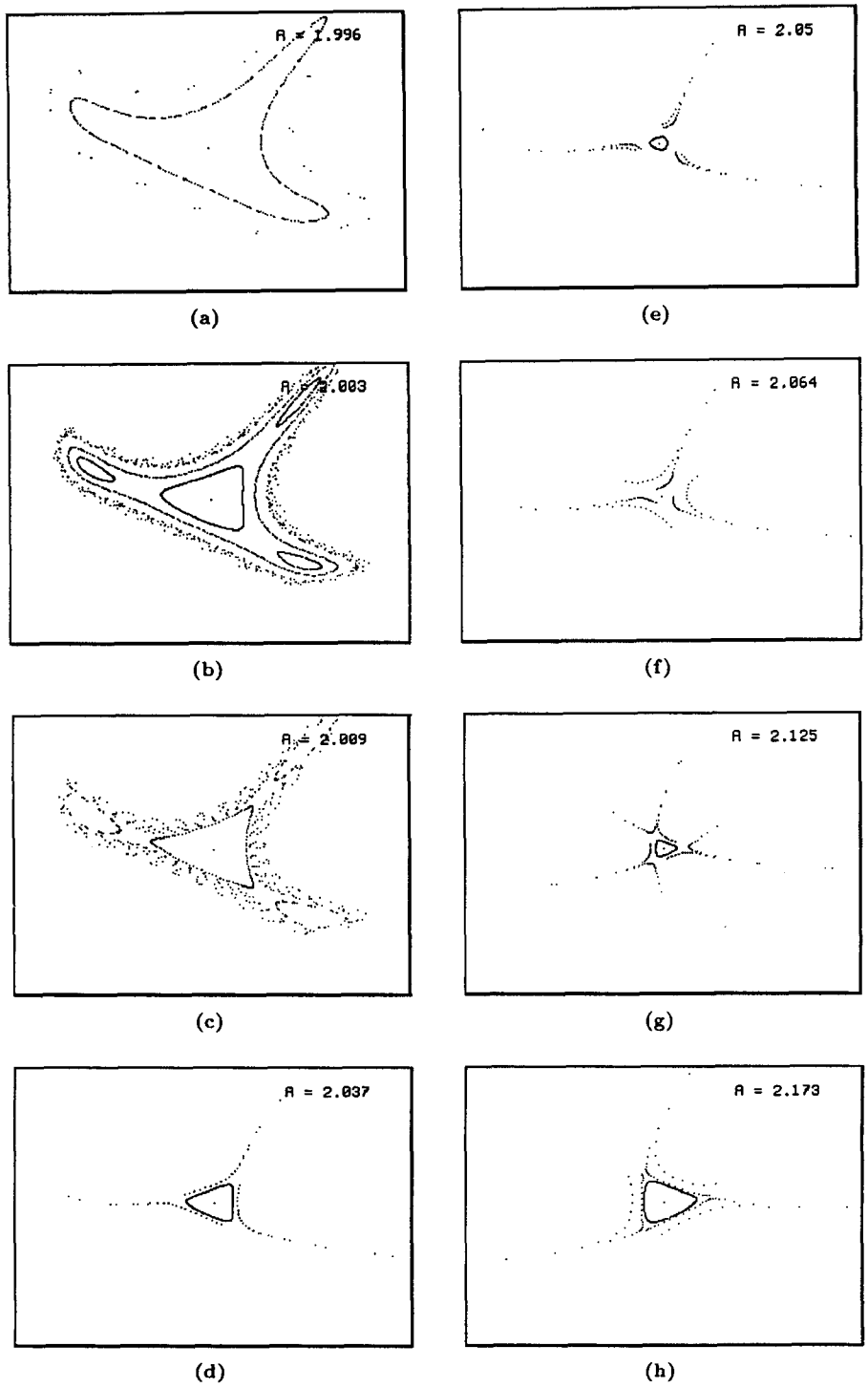


Fig. 25. Starting from two pairwise inconsistent phase portraits ((a) and (h)), KAM eventually finds a self-consistent sequence of phase portraits in the parameter range (1.996, 2.173) in accordance with the phantom-3-kiss bifurcation. KAM builds a global picture of behavior by piecing together theoretically acceptable sequences of bifurcations like this.

8. Conclusion

8.1. What does KAM do?

Given the governing equations of a one-parameter Hamiltonian system with two degrees of freedom, KAM automatically prepares numerical experiments, monitors the progress, interprets the results, and plans further experiments based on its current findings. The output is an executive report summarizing the major possible behaviors—bifurcation patterns and the extent of chaotic behavior—of the system as the parameter is varied. KAM rivals an expert dynamicist in its ability to make decisions about:

- how many iterates to look at;
- what initial conditions to choose;
- what parameter values to try.

8.2. Why does KAM work?

Viewed as abstract examples of AI technology, KAM is hardly novel. The uniqueness of the KAM program, and the source of its power, is that it uses classic AI methods to exploit specific domain knowledge based on rigorous mathematical results. Some of these traditional methods achieve new power when combined with deep knowledge of dynamical systems.

8.2.1. KAM's knowledge

KAM incorporates two types of knowledge: knowledge of physics and knowledge of geometry.

Knowledge of physics

Hamiltonian systems are very special dynamical systems: they preserve volume in phase space. Volume preservation has many consequences. Some of these consequences about two-degrees-of-freedom systems are embodied in KAM:

- There are four possible orbit types: periodic orbit (including fixed point), quasiperiodic orbit (including island chains), separatrix (or homoclinic orbit), and chaotic orbit (including escape orbit).
- Each orbit type has a distinct geometry on the Poincaré section: a periodic orbit appears as isolated points; a quasiperiodic orbit as a single closed curve; island chains as multiple isolated closed curves; a separatrix as multiple connected loops; and a chaotic orbit as a random splatter of points covering a region. Escape orbits will leave any bounded region in phase space.

	Physics	Geometry
Production rules	If two quasiperiodic orbits have incompatible rotation numbers then they are inconsistent	If the MST has a finite sequence of hyperbolic segments then it has a hyperbolic structure
Algorithms	Algorithm to compute the rotation number	Prim's algorithm to construct MST

Fig. 26. Examples of encoding knowledge in different forms.

- Orbits in phase space do not cross; consequently, on a two-dimensional Poincaré section, orbits surrounded by a quasiperiodic orbit will stay inside the quasiperiodic orbit forever.
- Orbits with incompatible rotation numbers cannot be adjacent to each other in phase space.
- Each island chain must enclose at least one periodic orbit. Similarly, each quasiperiodic orbit must enclose at least one fixed point.
- There are five types of generic bifurcations: (1) extremal, (2) period doubling, (3) phantom-3-kiss, (4) phantom-4-kiss, and (5) Poincaré -Birkhoff. For reversible mappings, the Rimmer bifurcation is also generic [14].

Knowledge of geometry

KAM knows some geometry—"know" in the sense of knowing how. This knowledge includes abilities to:

- recognize a closed curve;
- detect clusters in a point set;
- locate points of extremal curvature on a curve;
- describe the shape of a minimal spanning tree in symbolic terms;
- determine spatial relationships among orbits: adjacency and inside/outside relation;
- compute convex hull;
- find intersection of two convex polygons;
- estimate area enclosed by a closed curve.

8.2.2. How is knowledge embodied in KAM?

Knowledge is represented in one of the two ways: production rules or algorithms. Physics knowledge in terms of productions rules, physics knowledge in terms of algorithms, geometry knowledge in terms of production rules, and geometry knowledge in terms of algorithms—all these forms are stored in KAM. Figure 26 shows examples of each type of representation.

8.2.3. How is knowledge used?

Processing in KAM occurs in three levels: orbit, phase space, and parameter space. Each level is defined by an interpreter whose main activity is the iterative applications of three operations: aggregation, partition, and classification. The output of one level, an object with a semantic label, is fed into the next higher level. The basic units processed at each level are different. The interpreter on the first level operates on point sets, while interpreters on higher levels operate on larger units such as orbits (collection of point sets) and phase portraits (collection of orbits). At each level of processing, an interpreter extracts information that is more concise and more global than that contained in the input.

Knowledge of physics and geometry is integrated into the processing via the three operations. First, the aggregation operation requires an adjacency relation, which comes from knowledge of geometry, and a pairwise consistency relation, which comes from knowledge of physics. Second, the partition operation can be based on domain-independent geometric considerations (such as determining the inconsistent edges in a MST) or on domain-specific physical facts (such as grouping phase portraits into legal bifurcation patterns). Third, the classification operation is completely domain-dependent, relying on facts about physics.

8.3. What are the contributions?

8.3.1. Smart computers know what not to compute

Dynamics of a nonlinear system is complex, but is not arbitrary. There are constraints in the phase space and parameter space that restrict the class of legal trajectories and legal bifurcation patterns. These constraints enable KAM to infer a good understanding of a physical system from only a small, but well chosen, set of experiments. *Uncovering what these constraints are and articulating them in some formal language is a major contribution of this research.* These explicitly formulated constraints can be taught not only to a computer, but also to students of dynamics. (For a similar view, see [18].)

8.3.2. Computers, like people, need imagistic reasoning

That problem solvers employing visual or diagrammatic representations can be more effective than those relying on symbolic representations alone is hardly new. What is provocative, however, is the suggestion that our thought processes are importantly imagistic, and that visual thinking may play a crucial role in problem solving. This view can have important consequences. For instance, in scientific computation there has been tremendous emphasis on visualization, but this has mostly meant the development of computer graphics technology to aid *human* visualization [10]. I believe that imagistic

reasoning is a very general class of problem solving strategies, and that progress in scientific computation may depend on machines' ability to visualize as well as on people's.

References

[1] V.I. Arnold, *Mathematical Methods of Classical Mechanics* (Springer, Berlin, 1978).

[2] S. Baase, *Computer Algorithms* (Addison-Wesley, Reading, MA, 1978).

[3] H. Gelernter, Realization of a geometry-theorem proving machine, in: E.A. Feigenbaum and J.A. Feldman, eds., *Computers and Thought* (McGraw-Hill, New York, 1963).

[4] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields* (Springer, Berlin, 1983).

[5] M. Henon, Numerical study of quadratic area-preserving mappings, *Q. Appl. Math.* **27** (1969) 291–301.

[6] M. Henon and C. Heiles, The applicability of the third integral of motion: some numerical experiments, *Astron. J.* **69** (1964).

[7] W. Lee and B.J. Kuipers, Non-intersection of trajectories in qualitative phase space, in: *Proceedings AAAI-88*, St. Paul, MN (1988).

[8] A.J. Lichtenberg and M.A. Lieberman, *Regular and Stochastic Motion* (Springer, Berlin, 1983).

[9] R.S. MacKay, Renormalization in area preserving maps, Ph.D. Thesis, Princeton University, Princeton, NJ (1982).

[10] B. McCormick, T. Desanti and M. Brown, Visualization in scientific computing, *Comput. Graph.* **21** (6) (1987).

[11] K.R. Meyer, Generic bifurcations of periodic points, *Trans. Am. Math. Soc.* **149** (1970).

[12] A.J. Nevins, Plane geometry theorem proving using forward chaining, *Artif. Intell.* **6** (1975) 1–23.

[13] G. Novak, Representations of knowledge in a program for solving physics problems, in: *Proceedings IJCAI-77*, Cambridge, MA (1977).

[14] R. Rimmer, *Generic bifurcations for involutionary area preserving maps*, Memoirs of the American Mathematical Society **272** (American Mathematical Society, Providence, RI, 1983).

[15] E.P. Sacks, Automatic qualitative analysis of dynnamic systems using piecewise linear approximations, *Artif. Intell.* **41** (1990) 313–364.

[16] M. Stallman and G.J. Sussman, Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis, *Artif. Intell.* **9** (1977) 135–196.

[17] P. Struss, Global filters for qualitative behaviors, in: *Proceedings AAAI-88*, St. Paul, MN (1988).

[18] G.J. Sussman and R.M. Stallman, Heuristic techniques in computer-aided circuit analysis, *IEEE Trans. Circuits Syst.* **22** (11) (1975) 857–865.

[19] W. Tsai, D. Yue and K.M.-K. Yip, Resonantly excited regular and chaotic motions in a rectangular wave tank, *J. Fluid Mech.* **216** (1990) 343–380.

[20] D. Waltz, Understanding line drawings of scenes with shadows, in: P.H. Winston, ed., *The Psychology of Computer Vision* (McGraw-Hill, New York, 1975).

[21] A.P. Witkin, Scale-space filtering, in: *Proceedings IJCAI-83*, Karlsruhe, Germany (1983).

[22] K.M.-K. Yip, Generating global behavior using deep knowledge of local dynamics, in: *Proceedings AAAI-88*, St. Paul, MN (1988).

[23] K.M.-K. Yip, KAM: automatic planning and interpretation of numerical experiments using geometrical methods, AI-TR 1163, MIT, Cambridge, MA (1989).

[24] C.T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, *IEEE Trans. Comput.* **20** (1971).