

Technical Report 1163

# KAM: Automatic Planning and Interpretation of Numerical Experiments Using Geometrical Methods

Kenneth Man-Kam Yip

MIT Artificial Intelligence Laboratory

*This blank page was inserted to preserve pagination.*

# **KAM: Automatic Planning and Interpretation of Numerical Experiments Using Geometrical Methods**

by

Kenneth Man-kam Yip

Submitted to the Department of Electrical Engineering and Computer Science  
on August 28, 1989 in partial fulfillment of the requirements for the degree of  
Doctor of Philosophy in Computer Science and Engineering  
at the Massachusetts Institute of Technology

## **Abstract**

KAM is a computer program that can automatically plan, monitor, and interpret numerical experiments with Hamiltonian systems with two degrees of freedom. The program has recently helped solve an open problem in hydrodynamics – the prediction of onset of chaos in a resonantly excited rectangular wave tank of finite depth. Unlike other approaches to qualitative reasoning about physical system dynamics, KAM embodies a significant amount of knowledge about nonlinear dynamics. KAM's ability to control numerical experiments arises from the fact that it not only produces pictures for us to see, but also *looks at* (sic - in its mind's eye) the pictures it draws to guide its own actions. By combining techniques from computer vision with sophisticated dynamical invariants, KAM is able to exploit mathematical knowledge, represented in terms of a "grammar" that dictates consistency constraints on the structure of the phase space and parameter space. KAM is organized in three semantic levels: orbit recognition, phase space searching, and parameter space searching. 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.

**Keywords:** Artificial Intelligence, Scientific Computing, Scientific Visualization, Imagistic Reasoning, Knowledge-based Systems, Nonlinear Dynamics, Hamiltonian Mechanics, Numerical Experiments.

**Thesis Supervisors:** Harold Abelson and Gerald Jay Sussman

*This empty page was substituted for a  
blank page in the original document.*



## Acknowledgments

In working on this dissertation I have benefited greatly from the gentle encouragement, perceptive observations, and helpful suggestions of many people. Here are those who I would like to thank especially:

Gerry Sussman, my thesis supervisor, who shaped my intellectual life and taught me everything from picking locks, to giving talks, to articulating ideas that appear to be too vague or too complex to deal with, and to doing artificial intelligence research.

Harold Abelson, my thesis supervisor, who taught me about teaching, effective writing, and how to think like a mathematician.

Jack Wisdom, my thesis reader, who taught me Hamiltonian mechanics and how an expert physicist thinks.

Pete Szolovits, who helped me during my early years in the AI Lab.

Nancy Kopell, Daniel Goroff, and Dick Hall, who taught me Dynamical Systems Theory.

Mike Eisenberg, Franklyn Turbak, and Mitch Resnick, my friends, who taught me how to teach 6.001 and improved the presentation of my thesis immeasurably.

Rich Doyle and Feng Zhao, my officemates over the years, who made Room 823 and 438 a special place to work.

Chris Lindblad, Chris Hanson, and Andy Boughton, who helped me install a Symbolics in my office.

the other members of Project Mac, who made the fourth floor a fun and lively forum.

Shui-Chun Mui, Shui-Ching Lo, and the Mui's family for their love and support through all these years.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-86-K-0180 and in part by contract N00014-85-K-0124. While conducting this research, the author was supported by a 3-year fellowship from General Electric Company, Schenectady, New York.

*This empty page was substituted for a  
blank page in the original document.*

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	GOAL: AUTOMATE PART OF A SCIENTIST'S WORK . . . . .	4
1.2	HENON'S AREA-PRESERVING MAP: AN ILLUSTRATION . . . . .	6
1.3	WHAT KAM DOES . . . . .	8
1.4	HOW KAM DOES IT . . . . .	10
1.5	WHAT DOES KAM BUILD ON? . . . . .	13
1.6	REST OF THE STORY . . . . .	13
<b>2</b>	<b>Overview of KAM</b>	<b>15</b>
2.1	A SESSION WITH KAM . . . . .	15
2.2	DESIGN CRITERIA . . . . .	18
2.3	KAM OVERVIEW: STRUCTURE AND PROCESSES . . . . .	19
2.3.1	Major components of KAM . . . . .	19
2.3.2	The computational processes . . . . .	19

<b>3</b>	<b>Orbit Recognition</b>	<b>29</b>
3.1	WHAT IS ORBIT RECOGNITION? . . . . .	31
3.1.1	Determining topological dimension . . . . .	38
3.1.2	Sequencing points on a curve . . . . .	38
3.1.3	Detecting clusters . . . . .	39
3.1.4	Counting loops of a separatrix . . . . .	40
3.1.5	Deciding when steady state is reached . . . . .	40
3.2	CORE IDEA: SHAPE FROM MINIMAL SPANNING TREES . . . . .	41
3.3	SHAPE FROM MINIMAL SPANNING TREE . . . . .	52
3.3.1	Determining when iterates appear as a closed curve . . . . .	53
3.3.2	Partitioning iterates into distinct clusters . . . . .	54
3.3.3	Counting the number of loops of a separatrix . . . . .	54
3.3.4	Identifying Chaotic Orbits . . . . .	58
3.3.5	Determining steady state . . . . .	59
3.4	IMPLEMENTATION . . . . .	61
3.4.1	Computing the Minimal Spanning Tree . . . . .	61
3.4.2	Computing the diameter of MST . . . . .	61
3.4.3	Orbit Classification Rules . . . . .	62
3.4.4	Orbit Recognition Algorithm . . . . .	63
3.5	EXPERIMENT: MAIN ILLUSTRATION . . . . .	65
3.6	ANOTHER EXPERIMENT: ORBIT ON SURFACE OF A TORUS . . . . .	71
3.7	EVALUATING THE PERFORMANCE . . . . .	72
3.7.1	How reliable are the recognition results? . . . . .	72
3.7.2	What are the orbits that cannot be recognized? . . . . .	72
3.8	CONCLUSION . . . . .	75

<b>4</b>	<b>Phase Space Searching</b>	<b>79</b>
4.1	WHAT IS PHASE SPACE SEARCHING? . . . . .	80
4.2	CORE IDEA: CONSISTENCY OF NEIGHBORING ORBITS . . . . .	84
4.3	ROTATION NUMBER: CONTINUITY OF DISCRETE FLOW . . . . .	87
4.3.1	Rotation Number Defined . . . . .	87
4.3.2	Compatibility of Rotation Numbers . . . . .	89
4.4	ORBIT COMPATIBILITY . . . . .	92
4.4.1	Pairwise Orbit Consistency . . . . .	92
4.4.2	Single Orbit Consistency . . . . .	94
4.4.3	Suggestion Rules . . . . .	94
4.5	IMPLEMENTATION . . . . .	99
4.5.1	Representation of Phase Portrait . . . . .	99
4.5.2	Representation of Orbits . . . . .	99
4.5.3	Computing Rotation Number Bounds . . . . .	99
4.5.4	Consistency Complaints . . . . .	100
4.5.5	The Basic Algorithm: Consistency Maintenance . . . . .	100
4.5.6	Determining Adjacency . . . . .	103
4.6	EXPERIMENT: MAIN ILLUSTRATION . . . . .	103
4.7	OTHER EXPERIMENTS . . . . .	107
4.7.1	Phantom 3-Kiss Bifurcation . . . . .	107
4.7.2	Bak Map: Rimmer Bifurcation . . . . .	112
4.7.3	Standard Map: Toroidal Phase Space . . . . .	114

4.8	CONCISE PHASE PORTRAIT DESCRIPTION . . . . .	115
4.8.1	Identifying Primary Orbits . . . . .	117
4.8.2	Grouping Orbits . . . . .	118
4.8.3	Simplifying Orbit Structures . . . . .	119
4.8.4	Classifying Canonical Flow Patterns . . . . .	121
4.8.5	Determining Connectivity of Canonical Flow Patterns . . . . .	124
4.9	EVALUATING THE PERFORMANCE . . . . .	125
4.9.1	Missing Small Structures . . . . .	125
4.9.2	What Works Well . . . . .	126
4.9.3	What Does Not Work As Well . . . . .	128
4.10	CONCLUSION: THESIS RESTATED . . . . .	128
<b>5</b>	<b>Parameter Space Searching</b>	<b>131</b>
5.1	WHAT ARE BIFURCATIONS? . . . . .	132
5.2	CORE IDEA: STRUCTURAL STABILITY . . . . .	139
5.3	GENERIC BIFURCATIONS OF FIXED POINTS . . . . .	145
5.4	PHASE PORTRAIT COMPATIBILITY . . . . .	150
5.4.1	Pairwise Portrait Consistency . . . . .	151
5.4.2	Suggestion Rules . . . . .	155
5.5	IMPLEMENTATION . . . . .	158
5.5.1	Representation of a Phase Portrait . . . . .	159
5.5.2	Representation of a collection of Phase Portraits . . . . .	159
5.5.3	Consistency Complaints . . . . .	159

5.5.4	Basic Algorithm . . . . .	159
5.6	EXPERIMENT . . . . .	161
5.6.1	From $A = 1.328$ to $1.764$ . . . . .	161
5.6.2	From $A = 1.764$ to $1.982$ . . . . .	163
5.6.3	From $A = 1.982$ to $2.173$ . . . . .	164
5.7	OTHER EXPERIMENTS . . . . .	166
5.8	EVALUATING THE PERFORMANCE . . . . .	166
5.9	EXECUTIVE SUMMARY OF NUMERICAL EXPERIMENTS . . . . .	167
5.10	CONCLUSION: THESIS RESTATED . . . . .	172
<b>6</b>	<b>Solving an Open Problem</b>	<b>175</b>
6.1	SCENARIO . . . . .	176
6.2	STANDING WAVES IN A RECTANGULAR TANK . . . . .	178
6.3	MATHEMATICAL FORMULATION . . . . .	179
6.4	NUMERICAL FINDINGS . . . . .	187
6.4.1	Making a Poincaré section . . . . .	187
6.4.2	Compiling a description of Poincaré section into numerical procedures	188
6.4.3	Two interesting findings: banded and critical energy phenomena .	189
6.5	APPLYING AN ANALYTICAL APPROXIMATION METHOD . . . . .	190
6.5.1	The idea of Resonance Overlap . . . . .	190
6.5.2	Resonance overlap diagram . . . . .	193
6.5.3	Geometric algorithms to manipulate the resonance overlap diagram	194
6.6	COMPARING ANALYTICAL PREDICTIONS WITH NUMERICAL FINDINGS . .	198
6.7	HOW PHYSICAL PARAMETERS AFFECT CHAOS . . . . .	200
6.8	WHAT HAVE WE LEARNED? . . . . .	202

<b>7</b>	<b>Conclusion</b>	<b>206</b>
7.1	WHAT DOES KAM DO? . . . . .	206
7.2	WHY DOES KAM WORK? . . . . .	207
7.2.1	KAM's knowledge . . . . .	207
7.2.2	How is knowledge embodied in KAM? . . . . .	208
7.2.3	How is knowledge used? . . . . .	209
7.3	WHAT ARE THE CONTRIBUTIONS? . . . . .	209
7.3.1	Smart computers know what not to compute . . . . .	209
7.3.2	Computers, like people, need imagistic reasoning . . . . .	210
7.3.3	Intelligent systems are built by layers of specialized interpreters . . . . .	210
7.4	WHAT'S NEXT? . . . . .	211
7.4.1	Enhancing KAM . . . . .	211
7.4.2	Workbench Approach to Scientific Computing . . . . .	211
<b>A</b>	<b>User Manual</b>	<b>214</b>
A.1	KAM WINDOW . . . . .	214
A.2	TELLING KAM ABOUT THE MAPPING . . . . .	216
A.3	FOUR MODES OF KAM . . . . .	217



# Chapter 1

## Introduction

*Our present analytical methods seem unsuitable for the solution of the important problems arising in connection with non-linear partial differential equations and, in fact, with virtually all types of non-linear problems in pure mathematics. The truth of this statement is particularly striking in the field of fluid dynamics.... We could...justify our contention that many branches of both pure and applied mathematics are in general need of computing instruments to break the present stalemate created by the failure of the purely analytical approach to non-linear problems.*

- John von Neumann [31]

From the study of star and galaxy formation, to aircraft and wing design, to blood flow in the heart, and to microelectronics device modeling, scientists and engineers are confronted with nonlinear equations that are inaccessible to analytical methods. Nonlinear equations are full of surprises: nearby initial conditions can lead to widely different steady state behaviors, and changing the parameters of the equations slightly can result in abrupt transition from regular to chaotic behavior. However, our present analytical methods in mathematics are not adequate for the solution of most of these problems.

Over 40 years ago, John von Neumann already had the grand vision that some day high-speed computing devices might help scientists to “break the deadlock”, and achieve significant progress in many types of nonlinear problems. To some extent, computer scientists have been playing a key role in making von Neumann’s dream come true. They have been building faster and bigger computers. Our machines today are about six orders

of magnitude faster than those in von Neumann's days. The increase in memory size is equally impressive.

But why is it that almost all the key problems that von Neumann proposed as suitable for computational attacks still remain unsolved? Is this because our computers today are not fast enough? Not large enough? Perhaps so. But I think there is another reason: our present computers are too dumb. They are good at generating numbers or even pictures, but not at qualitative understanding. They can generate lots of low-level data, and they even know how to present them graphically for human comprehension. But they do not understand the numbers they generate nor the pictures they present. They do not know what the essential features of the solution are. They cannot represent the important features in such a way that they can talk about them, reason about them, and use them to guide further experiments.

My thesis is that in order to accomplish what von Neumann envisioned, our computers must be much smarter. They should be a helpful assistant to a scientist. They should have the capability to automate many phases of a numerical experiment that require substantial human effort and judgment. For instance, they should be able to plan numerical experiments, monitor the simulation, and interpret the numerical results in high-level, qualitative terms. They should produce a kind of executive summary for the scientist, giving him the information that he really needs to solve his problems. Working with such intelligent assistants, scientists can concentrate their efforts on theory formulations, testing new ideas, and the design of new experiments.

I have built a suite of computer programs that are smart enough to serve scientists and engineers as useful problem-solving partners in studying a certain class of nonlinear problems. These programs perform tasks that go beyond number crunching, algebraic manipulation, or graphical display. They proceed directly from a system of equations to a global, qualitative description of the possible long-term behaviors. They do not produce rigorous proofs of the solution; instead, they provide scientists with strong intuitions about the consequences of their mathematical models.

This dissertation is about a theory of problem solving embodied in this set of computer programs. The programs are collectively known as KAM.<sup>1</sup> KAM can autonomously explore the behaviors of a specified dynamical system: it plans and executes numerical experiments, and interprets the results of numerical computations in high-level qualitative terms.

---

<sup>1</sup>The acronym stands for three mathematicians – Andrei Kolmogorov, Vladimir Arnold, and Jürgen Moser – who proved a key theorem in modern classical mechanics.

KAM operates in the world of Hamiltonian (conservative) systems with two degrees of freedom. KAM knows about Hamiltonian dynamics, and knows how to recognize and manipulate two-dimensional shapes. Because of its knowledge of dynamics and geometry, KAM can visualize solutions to a dynamical system as concrete geometric objects. Qualitative, geometric reasoning about these geometric objects suffices to determine the gross, long-term characteristics of the system. Effects due to perturbation of initial states or different forcing functions are modeled by bending and distorting the shape of these geometric objects. KAM is the first implemented computational system that combines deep knowledge of Hamiltonian dynamics with techniques from computational vision.

This dissertation makes two claims. The first claim is that for computers to make a genuine impact in science, we must be able to capture and articulate deep domain knowledge in an explicit and precise manner so that computer programs can reason about it, manipulate it, and pass it on to other computer programs to deduce further consequences. In a sense this claim is related to an approach advocated by a subfield of artificial intelligence, the expert systems. But there is an essential difference. I am not making any claims about what the right computational tools are for capturing the knowledge. In particular, I do not think the present techniques in artificial intelligence – production systems, logic programming, semantic nets, for example – are adequate for many of the tasks that I want my programs to do. For instance, we still do not know how to represent shapes of most objects, 2D or 3D, so that they may be efficiently matched, manipulated, or learned.

A corollary of the first claim is that making intelligent computers for scientific computing means lots of hard work. The reason is that in many scientific disciplines the domain knowledge is not available in a readily digestible manner suitable for some so-called “knowledge engineer” to spoon-feed it to our computers. Often enough we will find domain experts having trouble in articulating certain aspects of their knowledge. I think there is good reason for it: they rarely think about their own thinking in explicit and unambiguous terms.

The second claim is that, like many mathematicians, a computer can also benefit from one of Poincaré’s great insights – the insight that the language of geometry and topology is a powerful way of thinking about dynamical systems. Just as thinking about the dynamics of linear time-invariant systems in terms of Fourier transforms is often simpler and more powerful than in terms of time-domain methods, describing nonlinear dynamics in a geometric language (including concepts such as topology, dimension, manifold, transversality, and so forth) provides greater insights than those obtained from time-series representation of solutions to the differential equations.

I believe there is much to be learned about geometric thinking – thinking about images, extracting information from them, and relating them to symbolic descriptions. But, to borrow a phrase from Seymour Papert, it’s hard to “think about thinking without actually thinking about thinking about something” [23]. I am proposing that the subject of dynamics is an incredibly rich domain for studying such thinking processes.

## 1.1 GOAL: AUTOMATE PART OF A SCIENTIST’S WORK

Formulating physical phenomena in precise mathematical language, working out the consequences of the formulation, and comparing how well the consequences actually predict experimental results – this is a scientist’s great task. Figure 1 shows the task pictorially. The task can be conveniently divided into four phases. The first phase, **modeling or theory formulation**, involves the construction of a mathematical model that captures certain aspects of the phenomenon being studied. Typically, a physical problem leads to a mathematical model composed of two parts: (a) a system of (ordinary or partial) differential equations, and (b) a set of boundary and/or initial conditions. Frequently, great skill and ingenuity are required to invent a model that is concise and realistic, and yet is simple enough to be amenable to mathematical analysis. The creation of a tractable mathematical model often depends on making simplifications and assumptions about the nature of the physical system and various external influences acting on it.

The second phase, **analysis**, concerns deducing properties of the mathematical model. Understanding the consequences of the mathematical model is often achieved by heuristic methods, involving the interplay between analysis, geometry, and numerical and symbolic computation.

The third phase, **measurement**, is the process of obtaining empirical data. The measurement process can be very time-consuming, requiring sophisticated instruments and techniques. In large experiments, it is common that the measurement is carried out by a special team of experimentalists.

The fourth phase, **model evaluation**, is marked by the theorist’s comparing the model predictions with experimental data obtained by direct observations of the real-world phenomenon. Ideally, solutions of the mathematical models have values that approximate the measured values in the physical system. If the match between the model predictions and empirical data is satisfactory, the scientist can write a paper summarizing his findings; otherwise, he goes back to the modeling step, modifying some of the old physical assumptions or adding new ones.

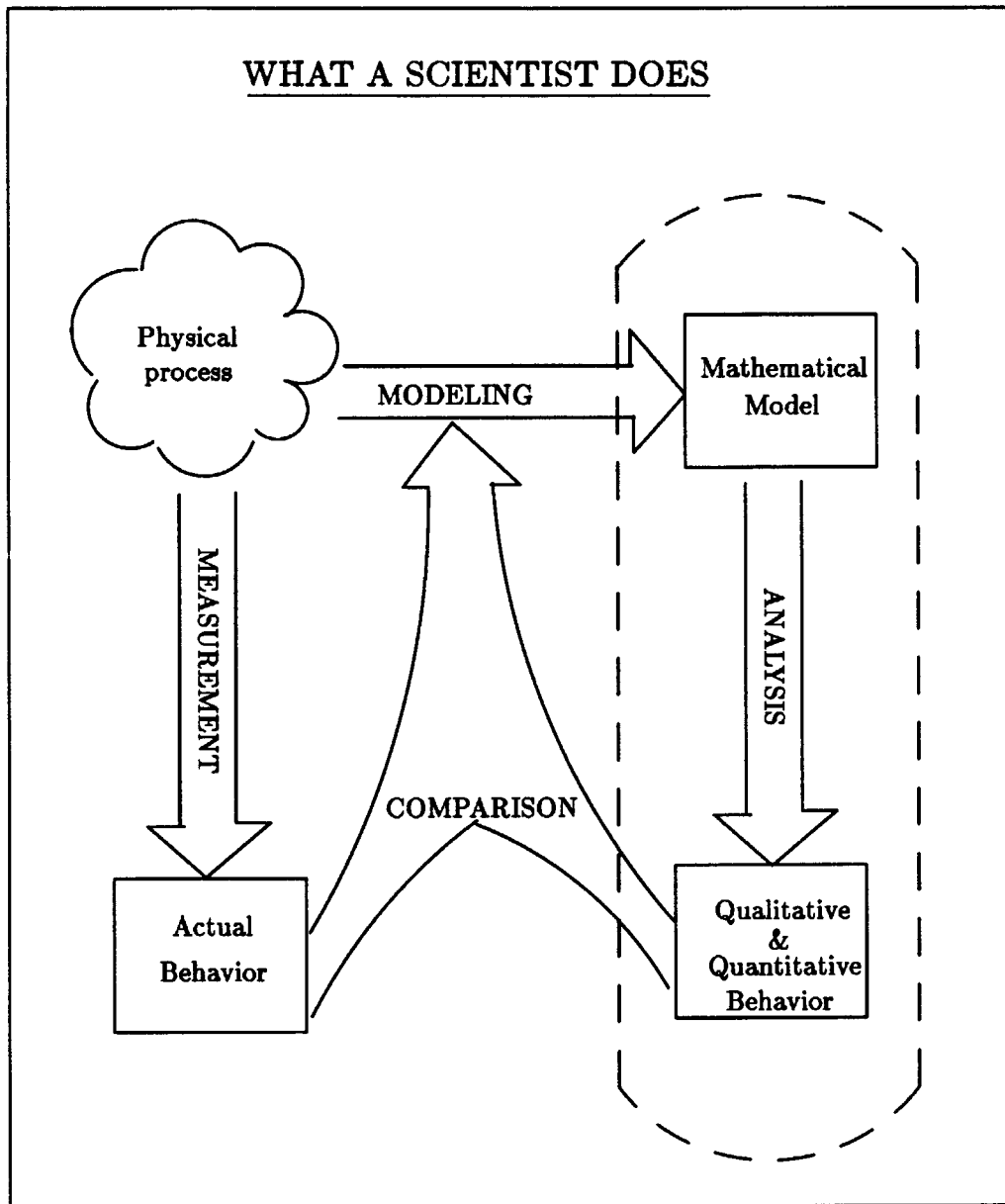


Figure 1.1: *Four phases of a scientific study: modeling, analysis, measurement, and model evaluation*

All four phases require substantial human effort, judgment, and insight. Nevertheless, in many cases – particularly when essential nonlinearity is present in the model – working out the consequences of the physical assumptions is much harder than formulating them.

A good example is the Navier-Stokes equations describing fluid motion. Knowing the governing equations sheds little light on the nature of turbulent motions of fluids.

Like Galileo's telescope which allows astronomers to track motion of distant planets, numerical simulation is an invaluable tool for revealing detailed consequences of a mathematical model. It is, however, a laborious process. Each particular set of initial states and parameter values generates one particular solution. To characterize the typical behavior of a system one must sample a large number of initial states and parameter values. Worse still, interesting behaviors often occur within a narrow region of phase space or parameter space. A brute force numerical exploration requires substantial computational time to generate numerical results, and substantial human effort to organize and comprehend them.

KAM is a prototype system that makes numerical exploration more tractable. The program makes an executive summary of the computation results in a form meaningful to a scientist, highlighting important qualitative properties of a system such as global structure of a phase portrait, bifurcations of steady states, and the fraction of phase space occupied by chaotic orbits as a function of parameter. It does not just present the numerical results graphically. It represents the essential features of the solution, reasons about it, and uses its understanding to guide selection of initial states and parameter values for further experiments.

## 1.2 HENON'S AREA-PRESERVING MAP: AN ILLUSTRATION

To illustrate how KAM works, we will use an example motivated by problems in celestial mechanics. The example is not really physical; it is a kind of mathematical abstraction, a simple stripped-down mathematical model. The purpose of the model is to illustrate the types of behavior that can arise in more complicated, realistic models.

In 1962 Michel Henon, a French astronomer, was thinking about motion of a star within a galaxy. In doing so, he was interested in a variety of questions. What kind of motion is possible for the star with different energy levels? For example, under what conditions will the star undergo simple periodic motion? More complicated recurrent motion? Or even chaotic motion? If the motion turns out to be chaotic, then it has interesting implications. For instance, it means predicting its place of origin is in all likelihood impossible because any small uncertainty of its current position and velocity will be greatly amplified when the motion is integrated forward or backward in time.

Henon assumed the gravitational potential of the galaxy is axisymmetric and so modeled the star motion as a Hamiltonian system – loosely, systems without friction – with two degrees of freedom. Even with great simplifications, the differential equations involved are still too hard to be solved analytically. So with the help of a graduate student, Carl Heiles, he numerically solved the equations for many different initial states and energy levels.

The conventional wisdom then was that only periodic or combinations of periodic motion are possible. After all, these are the kind of motion that classical analytical methods can handle. At low energies, Henon indeed found the regular orbits that classical theory predicts. But, to his surprise, when the energy was increased to a certain critical level, most of these regular orbits disappeared. In place of them were irregular, chaotic orbits that wandered around the phase space in a random manner. The phase portrait was filled with intriguing structures. Henon and Heiles could not come up with a mathematical explanation of what they observed.

Partly motivated by the intricacies of the star motion problem, Henon decided to look at a much simpler problem, which he hoped can lead to some mathematical insights. The simple model is subsequently known as the **Henon map**, and was published in an important paper “Numerical Study of Quadratic Area-preserving Mappings” in 1969 [11]. The model is extremely simple: just a pair of difference equations that map a plane onto itself.

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

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

The Henon map has a special property: area-preservation. That means if such a mapping is used to transform any region of the phase space, its area remain unchanged after the transformation. The area-preserving property is crucial because it is a consequence of Hamiltonian systems, systems that conserve energy. If you throw away area-preservation, you throw away the essential physics of a Hamiltonian system.

Without the quadratic terms, the Henon map describes a simple rigid rotation on the plane with an angle  $\alpha$ . But the quadratic terms are essential because they are responsible for creating a mixture of regular and chaotic orbits resembling those found in Henon and Heiles’ original problem.

Fig. 1.2 displays the result of a numerical experiment with the Henon Map for one particular parameter value  $\alpha$ . The picture is a mixture of a series of closed curves,

several island chains, and a region stippled with dots. The closed curves represent regular quasiperiodic motion, i.e., points move regularly around the curve. Points on the islands jump regularly from one island to another. The centers of the islands represent a stable periodic orbit that comes back to itself every 5 times. The stippled region represents chaotic orbits. Points belonging to these orbits hop around the phase space randomly. By adjusting the parameter  $\alpha$ , one can get a completely different picture with the regular and chaotic orbits mixed up in a different way.

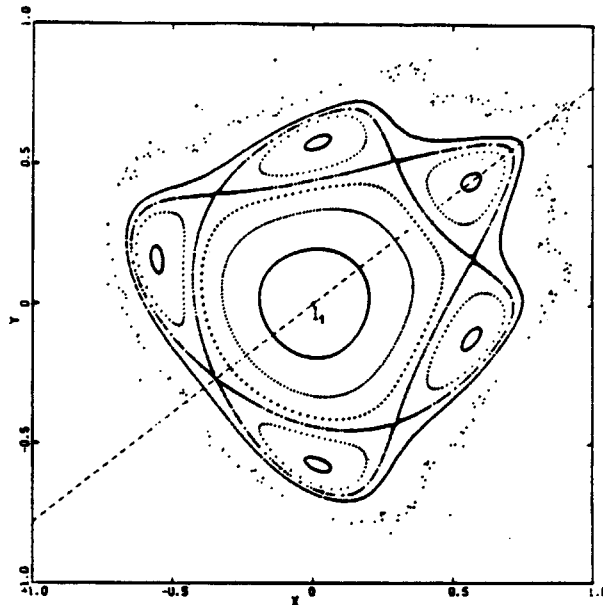


Figure 1.2: *The phase portrait for Henon map with  $\alpha = 1.3284305$ . This picture is directly taken from Henon's paper. Compare this with the one that KAM finds.*

Notice that Henon was very selective in performing the numerical experiment, and presenting the numerical results. He chose only a dozen orbits to represent the essential features of the phase portrait. There are lots of other initial states he did not try. Moreover, out of the infinitely many parameter values, he picked only ten of them to show the qualitative changes in the phase portraits as a function of parameter. How was Henon able to do this? Can a machine automate this experimenting process? Can it parse the visual arrangements of the points and generate meaningful interpretations? Can it produce summary descriptions like the ones that Henon gave in his paper?



### 1.3 WHAT KAM DOES

Fig. 1.3 displays the output of KAM when it is run on the Henon map with parameter value  $\alpha = 1.3284305$ . The upper panel shows the phase portrait that KAM finds. It consists of 10 orbits. The picture is remarkably similar to Fig. 1.2 taken directly from Henon's paper. The bottom panel shows KAM's summary of its finding in terms that are meaningful to dynamicists. Note that KAM does not have any natural language understanding capability. The description, written in stylized English, is solely for us to read. What's important is that KAM represents what it finds in some symbolic data structure that can be reasoned about and manipulated by other programs.

The example above represents a single experiment for one particular parameter value. After running a complete set of experiments, KAM produces 38 phase portraits corresponding to 38 different parameter values chosen from the range (0, 2.2). The phase portraits capture all the major bifurcations that occur in Henon map. What's interesting is that KAM finds something new: it finds a bifurcation of a large 7-island chain that is not reported in Henon's paper. Fig. 1.4 shows part of the executive summary that KAM generates. The summary contains the kind of high-level, qualitative information that a dynamicist may want to know in order to solve his problems.

### 1.4 HOW KAM DOES IT

Given a family of mappings depending on some parameter, how can one systematically explore its behavior by sampling a finite number of initial states and parameter values? Let's see how a human expert does this. The flow chart in Fig.1.5 illustrates the overall strategy a human experimenter may adopt in such exploration.

First of all, we must specify the input. It consists of three pieces of information: (1) the model equations, (2) the range of initial states, and (3) the range of some system parameter.

The first decision box is the termination condition for the experiment. The experimenter checks if he already has a self-consistent family of phase portraits. That is, he asks whether he has already found all the interesting qualitative changes in behavior as the parameter is varied. To illustrate the notion of qualitative changes, think of adding weights to the top of a vertical, elastic column. With a light weight, the column compresses a little. When the weight is gradually increased, the column suddenly buckles

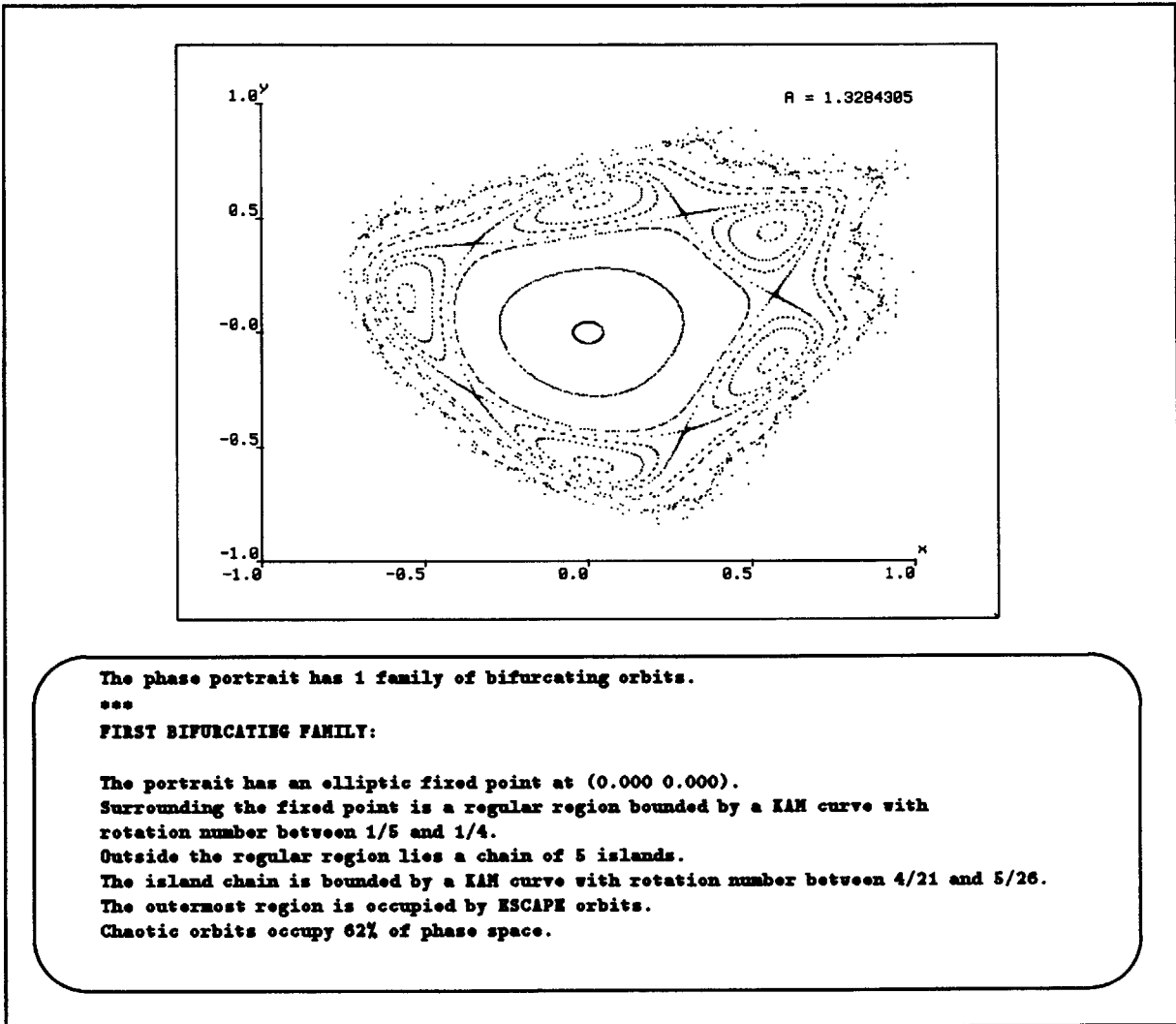


Figure 1.3: Upper: KAM produces the phase portrait for for the Henon map with  $\alpha = 1.3284305$  ( $\alpha$  is shown as "A" in the figure). Lower: KAM generates a summary description of the phase portrait.

to one side or the other. The vertical equilibrium position of the column has become unstable – a slight disturbance will move the column into a new stable state, one of the two buckled positions. So increasing the loading causes both the number of steady state behavior and their stability properties to change. In other words, changing the vertical loading causes a qualitative change in behavior of the column. We say a bifurcation has occurred.

Dynamicists like to represent behavior by orbits in a phase portrait. Qualitative

In the parameter range (0 2.2), KAM finds a total of 38 phase portraits. The following bifurcation patterns are observed:

**Poincare-Birkhoff bifurcation:**

A pair of period-8 orbits appears at about 0.875 and the 8-island chain disappears at about 1.075.

**Poincare-Birkhoff bifurcation:**

A pair of period-7 orbits appears at about 1.000 and the 7-island chain disappears at about 1.157.

**Poincare-Birkhoff bifurcation:**

A pair of period-6 orbits appears at about 1.075 and the 6-island chain disappears at about 1.257.

**Poincare-Birkhoff bifurcation:**

A pair of period-5 orbits appears at about 1.271 and the 5-island chain disappears at about 1.437.

**Poincare-Birkhoff bifurcation:**

A pair of period-4 orbits appears at about 1.574 and the 4-island chain disappears at about 1.655.

**Poincare-Birkhoff bifurcation:**

A pair of period-3 orbits appears at about 1.819 and the 3-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.

Figure 1.4: *KAM generates a summary description of the major bifurcation patterns for the Henon Map.*

changes in behavior corresponds to topological changes in the phase portraits. A self-consistent family of phase portraits just means that all the topological changes can be explained by some mathematical theory of bifurcation.

If a self-consistent family of phase portraits is found, then the experimenter summarizes his major findings from the experiments. Otherwise, he chooses another parameter value for further experiments. Of course, what parameter value to try next depends on his knowledge about what qualitative changes he has already found, and what kinds of bifurcation are likely to occur.

Behavior of a nonlinear system is in general very sensitive to the initial conditions. So the experimenter has to sample a sufficient number of initial states to map out the global behavior of the system. But the choice of these initial states are not random. It is suggested by what's seen so far, and often requires knowledge about how different pieces of behavior can be fitted together.

After he has chosen an initial state, he decides how long he should run the simulation. For a common type of simulation, namely, integrating differential equations, he has to decide how many integration steps to take for a given initial state. Taking too many steps is costly. On the other hand, taking too few steps will not reveal the steady state

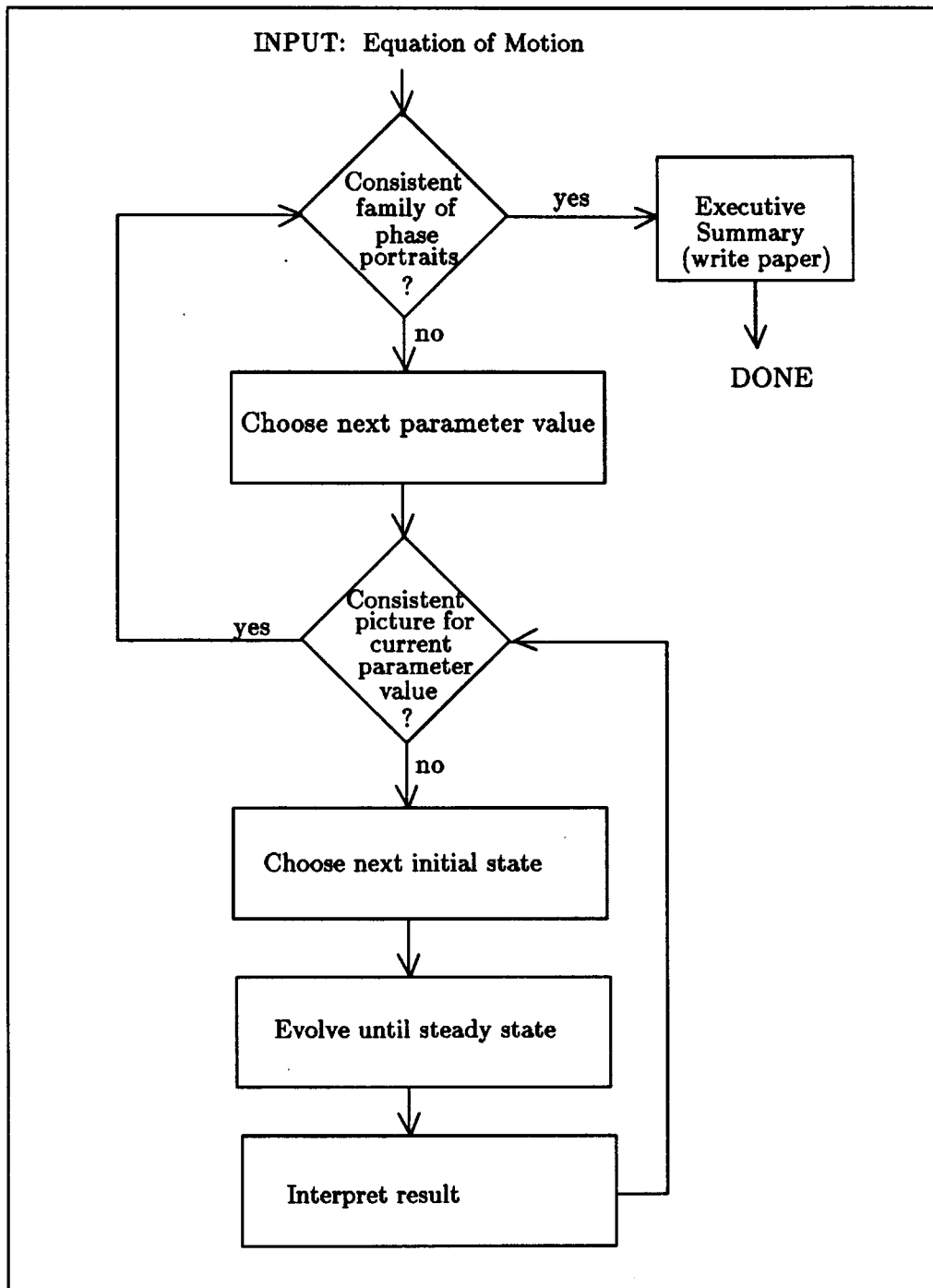


Figure 1.5: A flow chart showing the steps that a human expert may take in numerical experiments. Lots of knowledge and judgment are required in each decision box.

behavior. So he has to be able to make a judgment here.

Finally, he has to interpret the numerical results and extract the proper implications for empirical or further numerical verification.

I have just described how a human expert goes about doing numerical experiments to uncover consequences of some mathematical model. There are three identifiable pieces: (1) choosing parameter values and deciding when to stop, (2) choosing initial states, and (3) recognizing a steady state behavior. Each of these pieces requires decisions that depend on the expert's knowledge of what has been going on in the experiment, and his general knowledge about how such a mathematical model can behave.

KAM operates in an analogous way because its design is guided by the understanding of how I myself plan, monitor, and analyze numerical experiments. In particular, KAM uses a similar kind of knowledge – both mathematical and geometrical – to constrain its search for interesting behaviors as its human counterpart does.

## 1.5 WHAT DOES KAM BUILD ON?

A fundamental paradigm in building intelligent systems is the exploitation of physical constraints. The power of the paradigm is first demonstrated in David Waltz's program to interpret line drawings in a world of polyhedral objects [32].<sup>2</sup> The paradigm consists of three parts: (1) defining the primitive objects in the domain, (2) cataloging all legal combinations of the primitive objects, and (3) implementing a constraint satisfaction procedure based on the catalogue of legal combinations.

Waltz's methodology strongly influenced the design and implementation of KAM. On a certain level KAM is the same as Waltz's program: they capture significant physical constraints that dictate the structure of interactions among primitive objects of the domains; they use local evidence to arrive at global conclusions; and they embody the knowledge in terms of a constraint satisfaction algorithm.

---

<sup>2</sup>A few years before, David Huffman [13] and Maxwell Clowes [6] already had the idea that an exhaustive analysis of all possible local physical arrangements of surfaces and edges can provide a basis for solving the line interpretation problem. Waltz's contribution, however, was to demonstrate that a library of all legal junction geometries can lead to an *efficient* algorithm for interpreting line drawings.

## 1.6 REST OF THE STORY

The remainder of this dissertation is organized as follows. In Chapter 2, I give an overview of KAM. The first part of the overview describes what it is like to use KAM. The second part goes over the major pieces of the program. The next three chapters are concerned with each of these pieces.

Chapter 3 is about orbit recognition. This chapter explains the possible types of steady state behavior of a Hamiltonian system with two degrees of freedom. It then shows how KAM recognizes shape of orbits, and uses symbolic descriptions of shape to classify the type of an orbit.

Chapter 4 is about phase space searching. It explains the knowledge that KAM has about different ways in which orbits can fit together in phase space, and how this knowledge can be used to constrain its search for initial states.

Chapter 5 is about parameter space searching. It starts with a description of the bifurcation theory for Hamiltonian systems. It then describes how knowledge of typical bifurcations is embodied in KAM, and how such knowledge provides the basis for generating a high-level executive summary of the numerical experiments.

Chapter 6 is about using KAM to understand an open problem in hydrodynamics: the prediction of the onset of chaos in a wave tank.

The last chapter is the conclusion. I explain what really makes KAM work, describe how the state of the art is advanced by KAM, and sketch out some future directions to pursue.

# Chapter 2

## Overview of KAM

KAM is a collection of programs that can plan, monitor, and interpret numerical experiments involving Hamiltonian systems with two degrees of freedom. KAM embodies a significant amount of knowledge about nonlinear dynamics, and is able to visualize solutions to the dynamical systems as geometric objects. There are lots of details involved in explaining how KAM works. But there is little reason to get bogged down by the details now. The purpose of this chapter is to acquaint the reader with KAM: its use, its major pieces, and its underlying computational structure.

### 2.1 A SESSION WITH KAM

The following sample dialogue shows how KAM aids a scientist in exploring the dynamics of the Henon map. KAM is written in Zetalisp, running on the Symbolics Lisp Machine. 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. 2.1). 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  $\frac{5}{26}$  and  $\frac{1}{5}$ . It took 256 iterates to reach steady state (Fig. 2.2a).

**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  $\frac{4}{19}$  and  $\frac{3}{14}$ . It took 256 iterates to reach steady state (Fig. 2.2b).

**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. 2.3), 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 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.

**Scientist:** Show me all the major bifurcations as the parameter  $\alpha$  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:

**Poincare-Birkhoff bifurcation:**

A pair of period-8 orbits appears at about 0.875 and the 8-island chain disappears at about 1.075.

**Poincare-Birkhoff bifurcation:**

A pair of period-7 orbits appears at about 1.000 and the 7-island chain disappears at about 1.157.

**Poincare-Birkhoff bifurcation:**

A pair of period-6 orbits appears at about 1.075 and the 6-island chain disappears at about 1.257.

**Poincare-Birkhoff bifurcation:**

A pair of period-5 orbits appears at about 1.271 and the 5-island chain disappears at about 1.437.

**Poincare-Birkhoff bifurcation:**

A pair of period-4 orbits appears at about 1.574 and the 4-island chain disappears at about 1.655.

**Poincare-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 run a session with KAM. As shown in the dialogue, KAM has the following four capabilities:



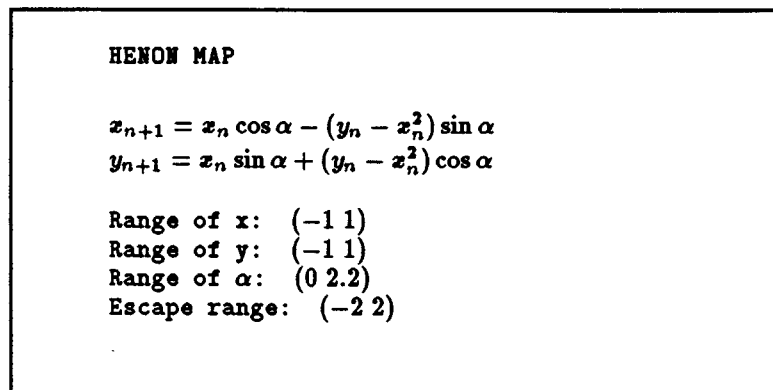


Figure 2.1: Input to KAM: (1) equations of motion, (2) range of state variables, (3) range of parameter, and (4) escape range.

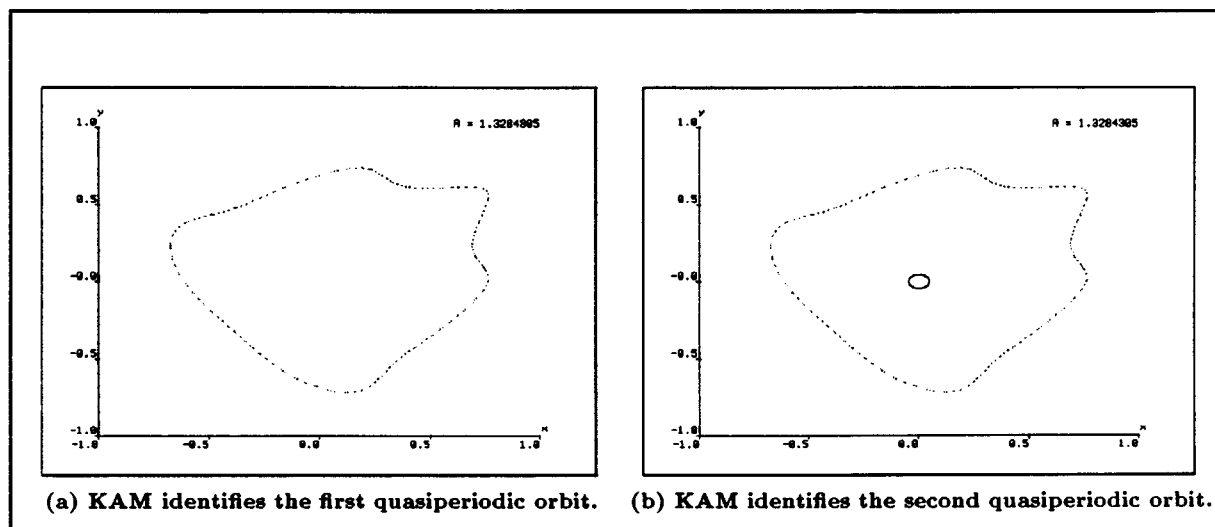


Figure 2.2: KAM decides how many iterates to look at. It also identifies the type of an orbit. If the orbit is quasiperiodic, KAM computes a bound for its rotation number.

- 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

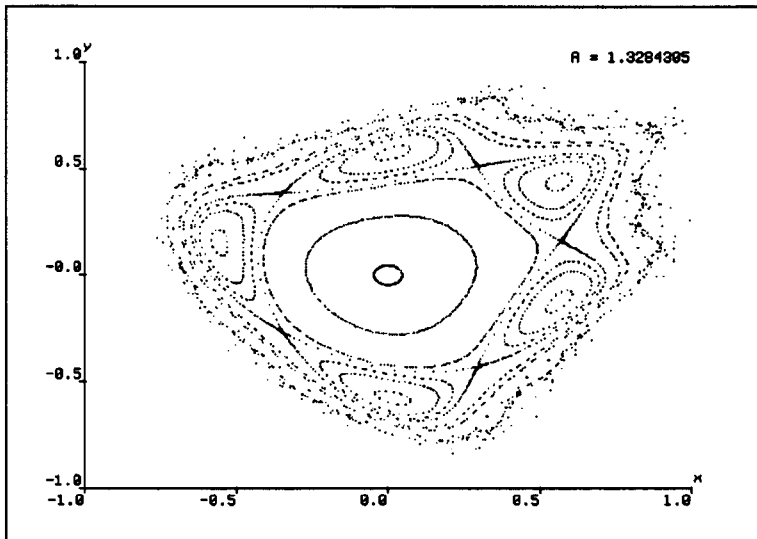


Figure 2.3: *KAM decides what initial states to pick. The phase portrait it produces consists of 10 orbits.*

about fixed point, orbit, rotation number, regular and chaotic region, and bifurcation.

The remainder of this chapter describes the major pieces of KAM, and the computational processes responsible for deriving a high-level interpretation from collections of discrete points in the phase space.

## 2.2 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 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.

The next section will explain how KAM is organized to meet these design criteria.

## **2.3 KAM OVERVIEW: STRUCTURE AND PROCESSES**

### **2.3.1 Major components of KAM**

Fig. 2.4 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 orbit 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.

### **2.3.2 The computational processes**

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

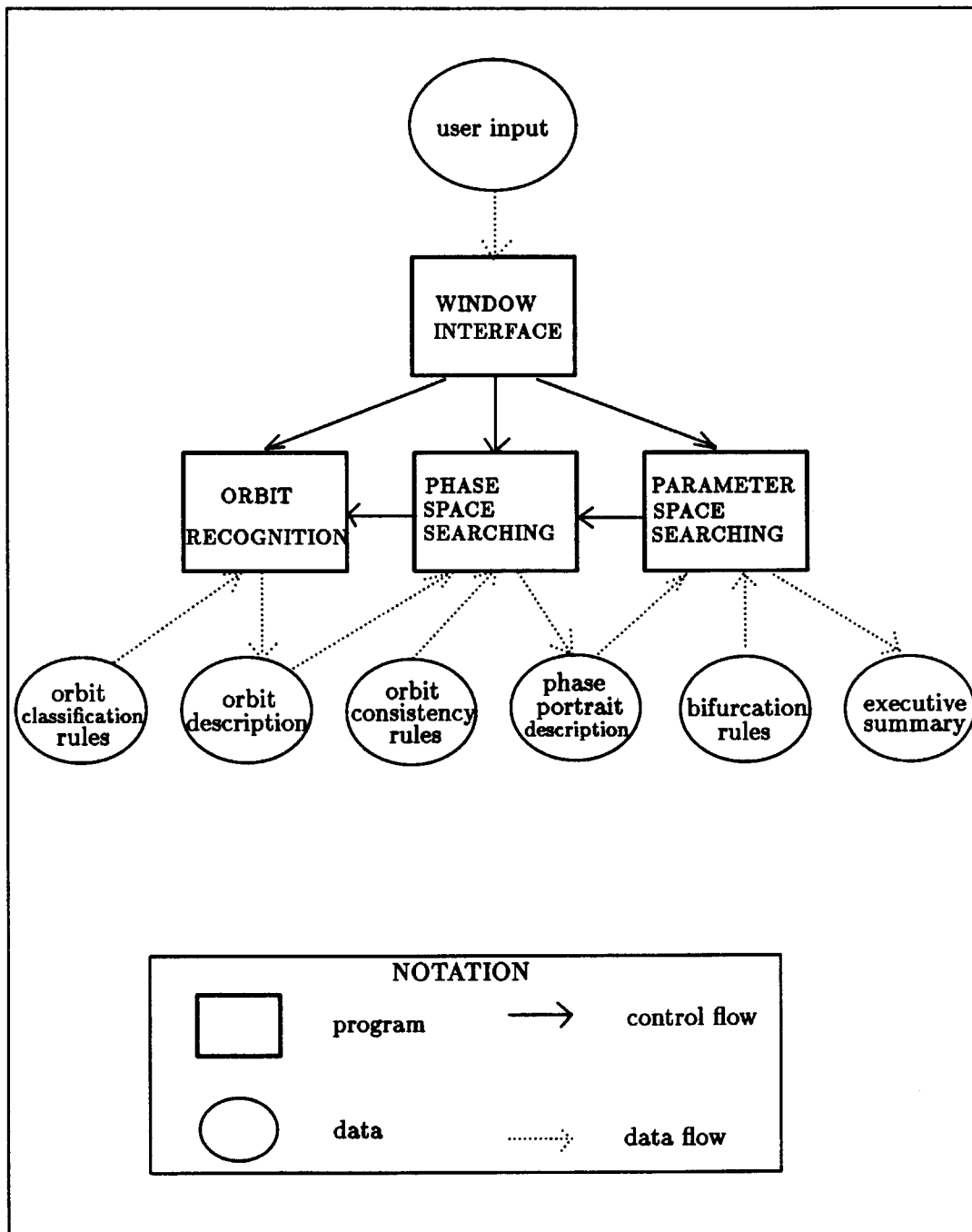


Figure 2.4: Major components of KAM.

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 2.5 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. The number of such levels depends on the domain semantics. 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 on the geometry of the point set. The recognition process proceeds in three steps (Fig. 2.6):

1. aggregating points into a minimal spanning tree

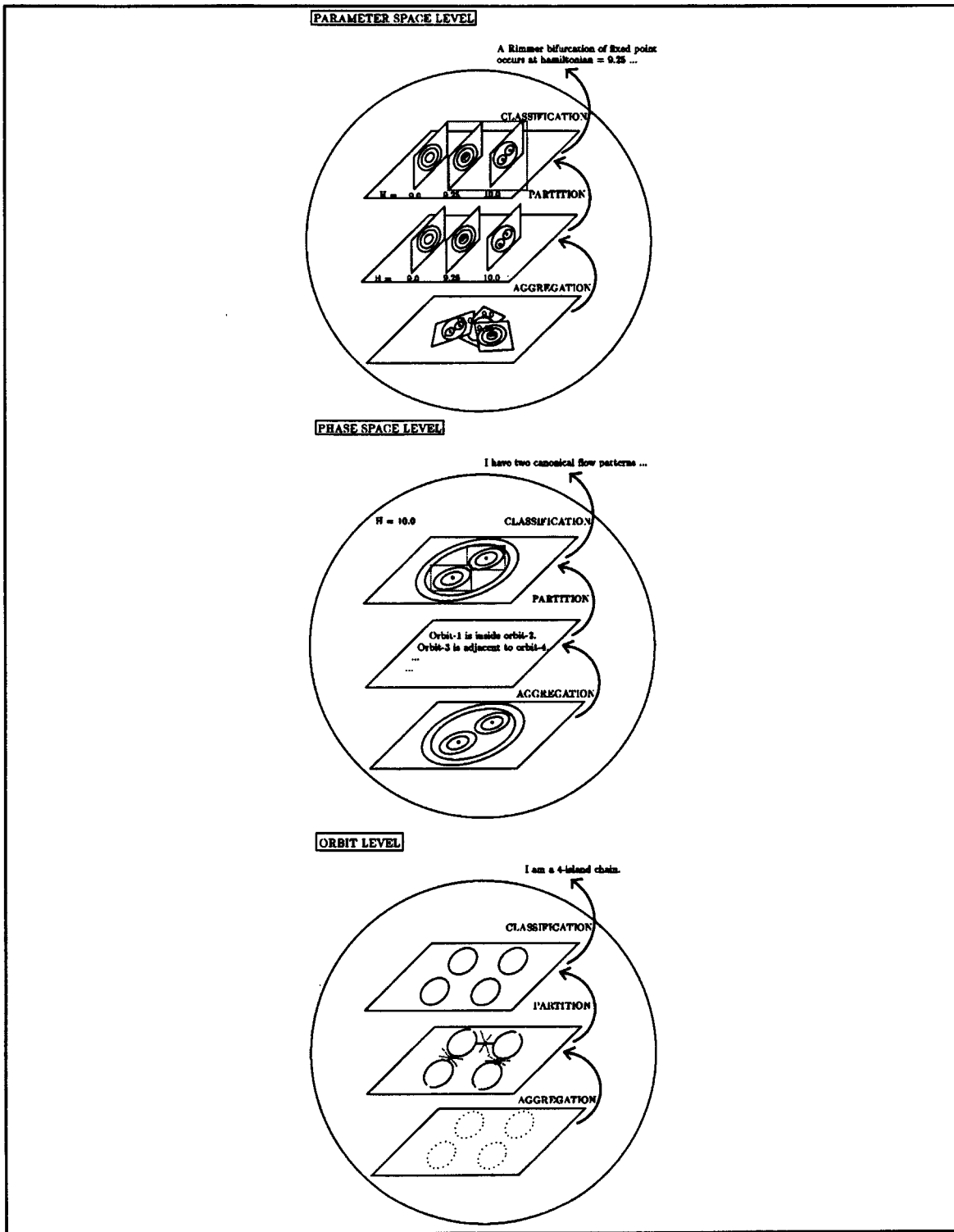


Figure 2.5: *Computational Structure of KAM. Focus on the shape of the processes. The same pattern of behavior – aggregation, partition, and classification – occurs in each of the three semantic levels.*

2. partitioning the minimal spanning tree into clusters
3. classifying the orbit

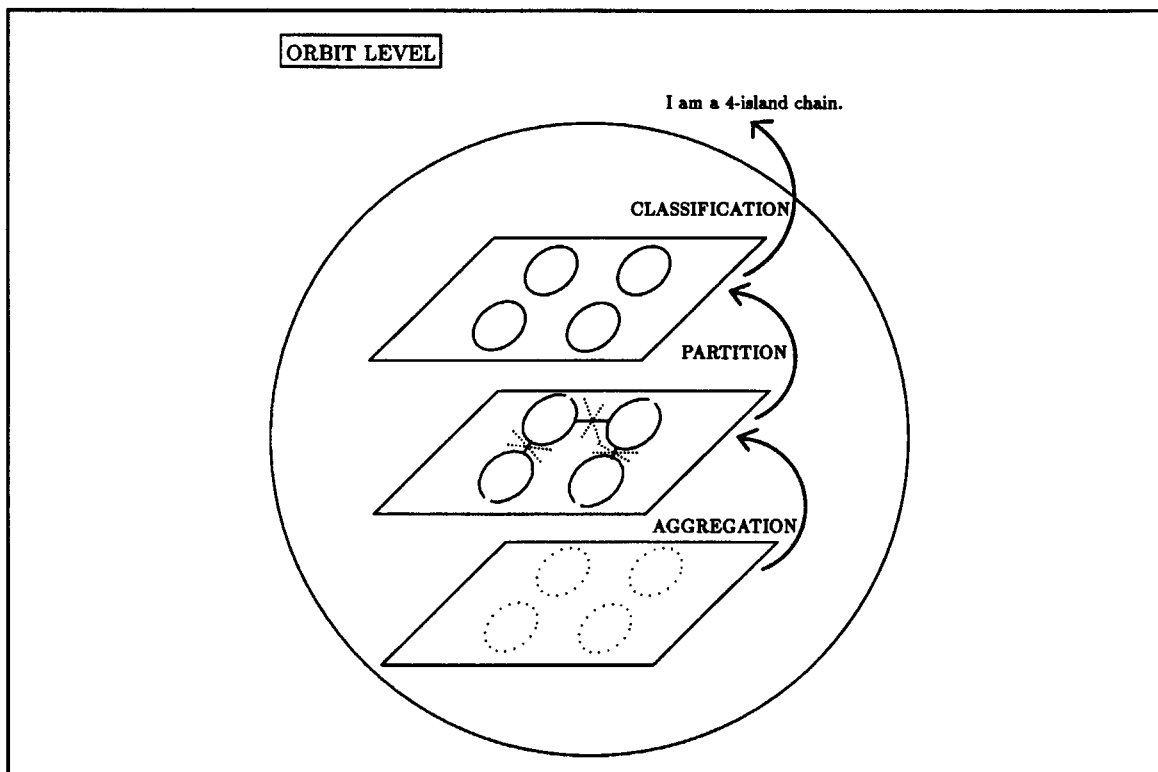


Figure 2.6: The collection of isolated points are aggregated in a MST. The MST is partitioned into clusters by deleting inconsistent edges. The inconsistent edges are marked by the crosses ( $\times$ ). The geometry and grouping structure of the point set are described symbolically. A point set is classified by matching its symbolic description with the symbolic descriptions of known orbit types.

**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 3 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. 2.6, 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. 2.7):

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



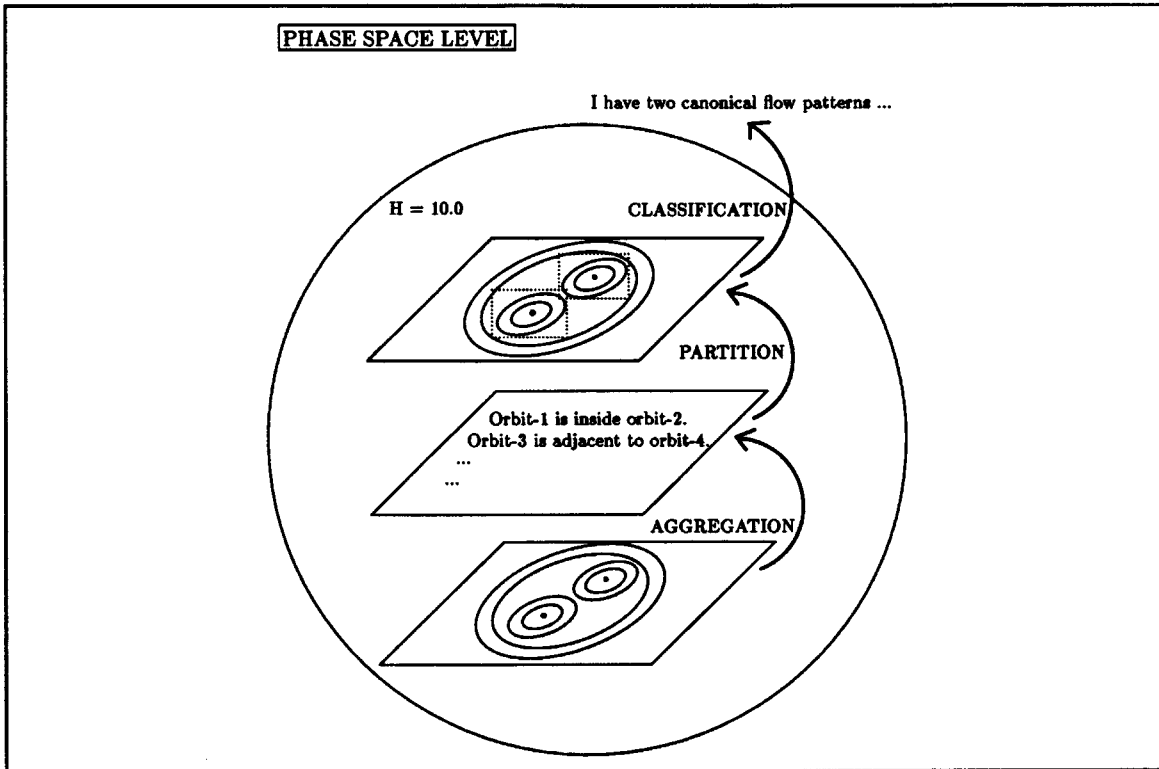


Figure 2.7: The collection of orbits are aggregated in an orbit adjacency network. The spatial relationships among orbits are symbolically described. The symbolic description is the basis for grouping orbits into canonical flow patterns. Each dotted box encloses a canonical flow pattern.

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 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 (section 4.8).

**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 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. 2.8):

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 1-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.

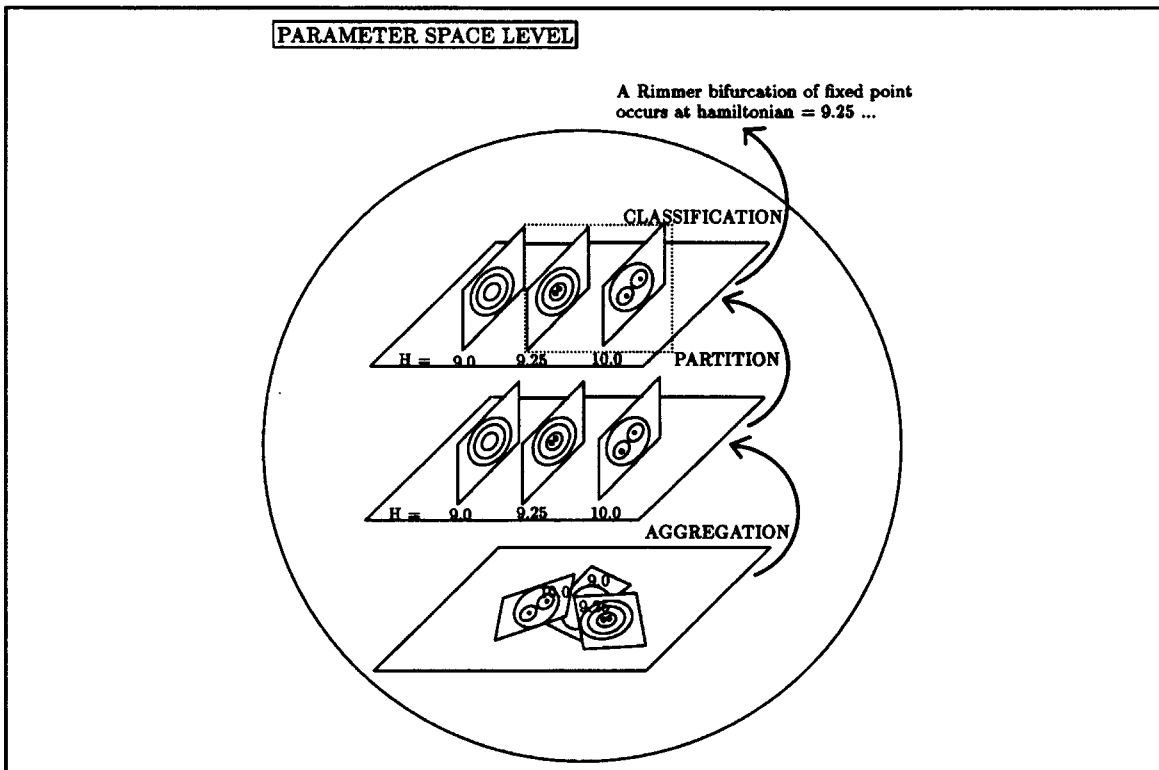


Figure 2.8: *The collection of phase portraits are ordered according to their parameter values. Topological changes between neighboring phase portraits are described symbolically. The symbolic description is matched against a list of known bifurcation patterns. The two phase portraits inside the dotted box are topologically equivalent.*

**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 lots of details in the description; they are left for the remaining chapters. 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.

# Chapter 3

## Orbit Recognition

Chapter 2 provided an overview of KAM's external behavior and its major pieces – orbit recognition, phase space searching, and parameter space searching. Now we are ready to analyze each of these pieces. This chapter and the next two are devoted to describing KAM's internal workings. The present chapter takes up the issue of how KAM classifies and recognizes different kinds of orbits. This is the **Orbit Recognition Problem**. In approaching the Orbit Recognition Problem two main questions arise:

- What are the rules for allocating a collection of iterates to one of the known orbit categories?
- How many iterates of an orbit are needed for a reliable classification?

Much of this chapter is concerned with expounding a computationally efficient solution to these problems. The remainder of this introduction briefly initiates that discussion.

Iterating a discrete map on a given initial point  $x$  generates a sequence of points  $x, f(x), f^2(x), \dots, f^n(x)$ . This sequence of points is called the **orbit of  $x$**  generated by the map  $f$ . An orbit in phase space describes how a system evolves from one state to another over an interval of time. Understanding the geometric properties (e.g., dimension, clustering, etc.) of an orbit suggests inferences about the qualitative behavior (e.g., chaotic wandering, periodic motion etc.) of the underlying dynamical system.

Any finite segment of an orbit can be thought of as a two-dimensional dot pattern. Even with little knowledge of dynamics, a naive observer is able to perceive structures

in the dot pattern. For instance, he can easily tell whether all the points seem to lie on some smooth curve, or whether they are grouped into distinct clusters. It seems that this ability to perceive shape or *gestalt* of a point set does not require specific domain knowledge – for example, knowledge about the purpose and function of the perceived structures. KAM attempts to mimic such perceptual organizational ability to discern meaningful orbit structures in a pattern of points.

KAM's domain-independent recognition method is based upon computing a geometric structure, known as the **minimal spanning tree (MST)**, of the points in the dot pattern. The MST aggregates isolated points into larger structures. By computing certain properties of the MST, one can extract shape information about the dot pattern, such as clustering and line structure.

The recognition method based on the MST is embodied in KAM's recognition module, which has two nested components: the **identification routine** and the **stability-control routine**. The identification routine constructs the MST of a dot pattern, computes relevant geometric properties of the tree structure, and massages the quantitative information into a qualitative symbolic description of the shape of the dot pattern. The symbolic shape description is then fed into a list of orbit identification rules that decides the orbit category. At this point, the stability-control routine takes over. Its purpose is to ensure robustness in the identification result. It calls the identification routine with a larger dot pattern (by taking a longer orbit segment). If the larger dot pattern is classified into the same orbit category, then the stability-control routine assumes the pattern has reached a steady state, and so it accepts the result of the identification routine. Otherwise, it will examine a still longer orbit segment and repeat the recognition process until a steady state is reached.

The content of this chapter is as follows. The first section introduces the problem of orbit recognition. Section 3.2 argues that the MST provides a sound basis for a domain-independent method of orbit recognition. Section 3.3 describes methods to compute shape of a dot pattern from its MST representation. Section 3.4 covers the implementation of the recognition algorithm. The next two sections, 3.5 and 3.6, present experimental results. The chapter concludes with a evaluation of the recognition algorithm.

### 3.1 WHAT IS ORBIT RECOGNITION?

The iterates of an initial point generated by a discrete map form a set of isolated points – the orbit through the initial point. Orbit recognition is the problem of classifying this orbit. Recognizing the orbit type is important because KAM must interpret what it “sees” in order to direct the course of its numerical experiments. This section describes the problems involved in orbit recognition. We begin, however, with a few remarks about the significance of these orbits and about the possible orbit types.

#### Types of orbits

Consider a system of many bodies moving under the influence of force law. Suppose that, at some initial time, the system is in some particular state, and that some time later, the system returns to exactly the same state, that is, all the bodies have exactly the same positions and velocities as before. We say then the system repeats itself. By the uniqueness of solutions to differential equations, if the system repeats itself once, then it will repeat over and over again. The motion is said to be **periodic**.

Periodicity is an extremely important concept: it provides many insights into the behavior of the system. Think of the Earth revolving around the Sun. If the motion of the Earth were genuinely periodic, then we would know for certain that it won't ever fall into the Sun, nor will it wander off into outer space.

A fundamental idea in dynamical systems theory is a method, due to Poincaré, for locating periodic solutions to systems of ordinary differential equations [9]. 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. Mathematicians give 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. So to look for a periodic solution, all we have to do is to search for orbits that are closed curves. Notice that this question of whether an orbit is a *closed curve* is a purely topological one; it has nothing to do with the size and shape of the 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 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: a  $(n-1)$ -dimensional hypersurface embedded inside a  $n$ -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 also be determined from its Poincaré mapping.

In classical Hamiltonian systems – think of them as systems with no friction – there is another type of motion that is quite common. This type of motion is called **quasiperiodic** or **almost-periodic**. Here we have several periodic motions with independent frequencies combined together. Consider two oscillators. Each of these oscillators exhibits a periodic motion. If the two periods are **commensurate**, i.e., they are both integer multiples of some number, then the combined motion is also periodic. The period of the combined system is actually the least common multiple of the two independent periods.

But if the periods are not commensurate, then the combined motion never repeats itself exactly even though it may come arbitrarily close to its starting point. This is why



we call the motion *quasiperiodic*.

To see what a **quasiperiodic orbit** looks like in phase space, we must first figure out what the phase space for the combined system is. Since a periodic motion corresponds to a circle (actually a closed curve, but we have said size and shape of the curve don't matter) in phase space, the space of two combined periodic motion can be thought as the product of two circles – a 2-torus. Topologically a quasiperiodic motion is pictured as an orbit winding around the surface of the torus (Fig. 3.1). If the periods are incommensurate, the orbit will cover the toroidal surface densely. You can imagine the entire phase space is filled with toroidal surfaces, one nested inside the other, each of which corresponds to a different ratio of the two periods.

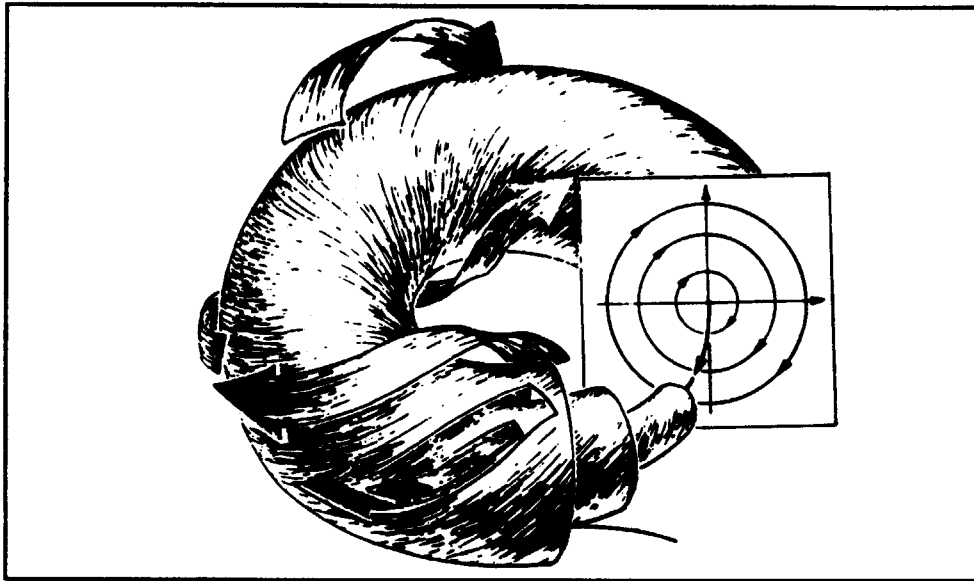


Figure 3.1: Classical picture of a Poincaré section near a periodic orbit. A quasiperiodic orbit is obtained by combining two incommensurate frequencies. Topologically, it covers the surface of a torus densely. Each circle on the section represents a quasiperiodic motion with two distinct periods. (From Stewart 1989)

This picture is simple and elegant: it says orbits of any Hamiltonian systems are either periodic or made up by a combination of periodic ones (Fig. 3.1). It is pretty much what classical dynamicists before Poincaré had in mind. Unfortunately it is wrong except for a small class of very special systems, called **integrable systems**. Integrable systems are those systems that can be solved explicitly by a formula. But such systems are extremely rare.

What's wrong with the classical picture is the assumption that the two oscillators that make up phase space are completely independent of each other. In most systems, they are coupled. And coupling can produce dramatic effects. An example of coupling effect was provided by Christian Huyghens, a 17th century scientist, who observed that two clocks hung on the same wall tend to synchronize. This phenomenon is known as **frequency entrainment**, and is caused by the nonlinear coupling between the two clocks.

How does nonlinear coupling change the orbits in phase space? One way to visualize what happens is to imagine that the effect of the coupling is equivalent to adding some sort of perturbation to the orbits. Such perturbation may cause a change the direction of a quasiperiodic orbit so that it no longer lives on one toroidal surface, but wanders freely from one surface to another. Think of the planets in the Solar System. A planet that used to be repeating or almost repeating its position may wander off, visiting almost every place inside the Solar System. Its long-term behavior will become very unpredictable. This new type of behavior is called **chaotic**.

But then why isn't our Solar System – a bona fide nonlinearly coupled many-oscillators system – full of planets that wander around in a random manner? Unfortunately we don't have a definite answer to this question yet. Perhaps after many eons most of the planets do become chaotic. But we don't know for sure.

What we do know is that under *very small* perturbation to a Hamiltonian system of uncoupled oscillators most of the classical quasiperiodic orbits still survive. That means if you randomly pick an initial setup, the chance of your getting one of those regular, well-behaved orbits is high, but there is always a small, but *non-zero*, chance that the resulting orbit is chaotic. This is a non-trivial assertion, and its proof involves a very difficult mathematical theorem – the celebrated KAM theorem, after the three famous mathematicians: Andrei Kolmogorov, Vladimir Arnold, and Jurgen Moser [14, 2, 28]. The tori on which these surviving quasiperiodic orbits live are often called **KAM tori**.

The KAM theorem, however, does not give us the whole story. For one thing it does not tell us what happens to the **resonant surfaces**, i.e., those tori inhabited by orbits that are combinations of commensurate periods. Do these “secondary” periodic orbits survive?

I will come back to this question later in Chapter 4 and Chapter 5. For now it suffices to say at least one pair of periodic orbits survive on each resonant surface. One of these periodic orbits is stable, and the other is unstable. The stable periodic orbit is itself wrapped around by quasiperiodic orbits. The structure around these stable

secondary periodic orbit resembles that around the primary periodic orbit: a kind of self-similarity. As a result we have secondary tori, which are the remnants of the breaking up of the resonant surfaces, winding around the primary torus. In between these secondary tori are tangled, chaotic orbits, known as **homoclinic orbits**, which connect up the unstable periodic orbit. The real picture is extremely complicated: all these periodic, quasiperiodic, and chaotic orbits are completely intertwined and mixed up (Fig. 3.2).

### **What orbits look like on the Poincaré section**

The modern picture that describes the phase space of two coupled oscillators is exceedingly complicated. It is very hard to visualize these three-dimensional structures winding around in space. We will see the Poincaré section gives the simplification that we need to get a handle on this problem.

Periodic, quasiperiodic, secondary periodic, homoclinic, and chaotic orbits – these form the complete list of orbits possible in Hamiltonian systems. What do they look like on a Poincaré section near some periodic orbit?

A periodic orbit is easy. If it goes around the torus once before it comes back to the exact starting point, then it corresponds to a single point on the section. We call this point a **fixed point** of the Poincaré mapping. The point is called “fixed” because as far as the mapping is concerned the point is unchanged even the periodic orbit itself is moving continuously all the time. If the periodic orbit goes around  $n$  times before it comes back to the same starting point, then it corresponds to a set of  $n$  points on the section. These  $n$  points are called **periodic points**, or a **periodic orbit**, of the Poincaré mapping. Note it may be a little confusing to use the term “periodic orbit” to refer to both the actual continuous periodic orbit in the phase space, and the set of isolated points that appear on the section. Anyway, the terminology is quite established in the dynamics literature. The periodic points corresponding to a stable periodic orbit are called **elliptic points**; the unstable ones are called **hyperbolic**.

Next let's turn to a quasiperiodic orbit. Remember a quasiperiodic orbit is an orbit that winds densely around the surface of a torus. On the section, it appears as a circle. More generally, a nested sequence of tori will show up as a nested sequence of concentric circles on the section. Again, mathematicians give these circles the same name as their continuous counterpart: **quasiperiodic orbits** or **almost-periodic orbits**. Yet another name is **KAM curves**, after the mathematicians who proved the existence of these orbits.

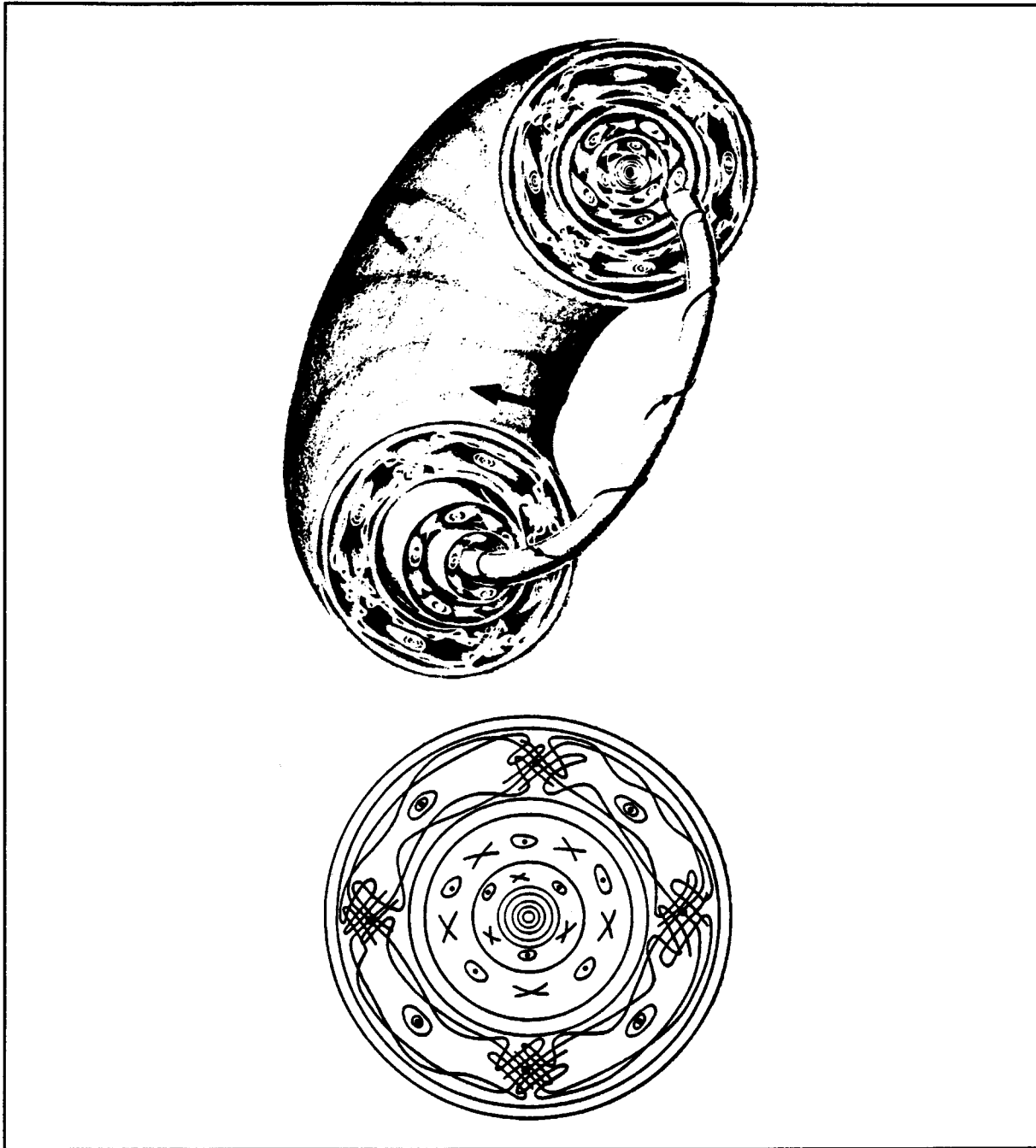


Figure 3.2: *Upper: What really happens near a typical periodic orbit: the Vague Attractor of Kolmogorov. Only some of the classical quasiperiodic orbits survive. Elsewhere, homoclinic orbits wind between resonance islands. (From: Abraham and Marsden 1978). Lower: Poincaré's schematic picture of orbits on a Poincaré section (first drawn by V. Melnikov).*

The third category is the secondary periodic orbits and quasiperiodic orbits around them. They appear as island chains on the section. The center of each island corresponds to one of the periodic points. Concentric circles around center are the quasiperiodic orbits.

Homoclinic orbits belong to the fourth category. A homoclinic orbit is a special type of chaotic orbit. It is chaotic because it does not stay on the surface of one torus. It is special because unlike a typical chaotic orbit it is confined to a small portion of phase space bounded by KAM tori. What this means is that even the motion of the orbit is not completely predictable, it can always be found in a limited region of the phase space. On the Poincaré section, it appears as a curve connecting up the hyperbolic periodic points. It is called a **homoclinic orbit** or sometimes a **separatrix**, by analogy with the separatrix of a one-dimensional pendulum.

The fifth category is the chaotic orbits. They are orbits that wander freely in a large region of phase space. On the section, they appear as points hopping around at random from one position to another. So stippled regions on the section indicate chaotic motion.

One final type of motion completes the list: unbounded motion. Orbits that are not confined in any finite volume of phase space are **escape orbits**. For an example, think of a meteor flying by the Earth and never comes back. On the Poincaré section, an escape orbit is a set of points that will eventually leave the section under the action of the Poincaré mapping.

### **Problems in recognizing the type of orbit**

Now that we have enumerated all the possible types of orbits as they appear on the Poincaré section, we should give precise geometrical criteria to characterize them. But it is difficult to come up with such criteria. There are two reasons for this. The first reason is that the distinction between orbit types depends on the behavior of an orbit in the asymptotic, infinite time limit. For instance, on the surface of section, a periodic orbit is defined as an orbit that is a superposition of commensurate frequencies; it closes on itself after a finite number of times around the torus. A quasiperiodic orbit is one obtained from incommensurate frequencies; it covers a topological circle densely. But this distinction between periodic and quasiperiodic orbit is a property of an infinite time limit. In other words, finite orbit segments are all topologically similar. Periodic orbits that have long period tend to cover a topological circle quite well, and they are therefore

indistinguishable from a quasiperiodic orbit if we just look at a finite number of iterates on the surface of section.

The second reason has to do with the problem of finite precision computer arithmetic. Round-off errors will prevent us from following an orbit exactly. So, for instance, we may see the initial segment of an orbit seem to fill a topological circle, but after a while the iterates spread. What appears as a smooth thin curve before may turn into a “fat” curve made up by iterates from several nearby orbits.

Because of these difficulties, it is clear that KAM cannot *prove* that a given orbit is an island chain or a separatrix or what not just by looking at a finite number of iterates. Rather, it provides some numerical evidence about the type of an orbit. With this caveat given, I shall start the discussion by first enumerating the tasks that arise in classifying orbits.

### **3.1.1 Determining topological dimension**

When are dots perceived as lying on a single continuous curve as shown in Fig. 3.3a?. Under what conditions do we perceive them as occupying a region in the two-dimensional space as in Fig. 3.3b? Answering these questions is important because it gives KAM a method to distinguish a chaotic orbit.

### **3.1.2 Sequencing points on a curve**

We tend to perceive the dot pattern in Fig. 3.4a as forming the boundary of a slightly distorted triangle. This organization dominates other possible organizations such as those shown in Fig. 3.4b and Fig. 3.4c. To see the dots as an outline of an object, we must be able to see these dots as connected in an ordered sequence. The computational question is this: given a set of dots that appear to lie on a curve, how can the dots be ordered? Ordering dots in a sequence is useful because it allows us to extract geometric information, such as curvature, of the object boundary.

### **3.1.3 Detecting clusters**

Looking at Fig. 3.5, we can see the dots fall naturally into 4 distinct clusters. How can the cluster structure in a dot pattern be detected? This question is related to the problem

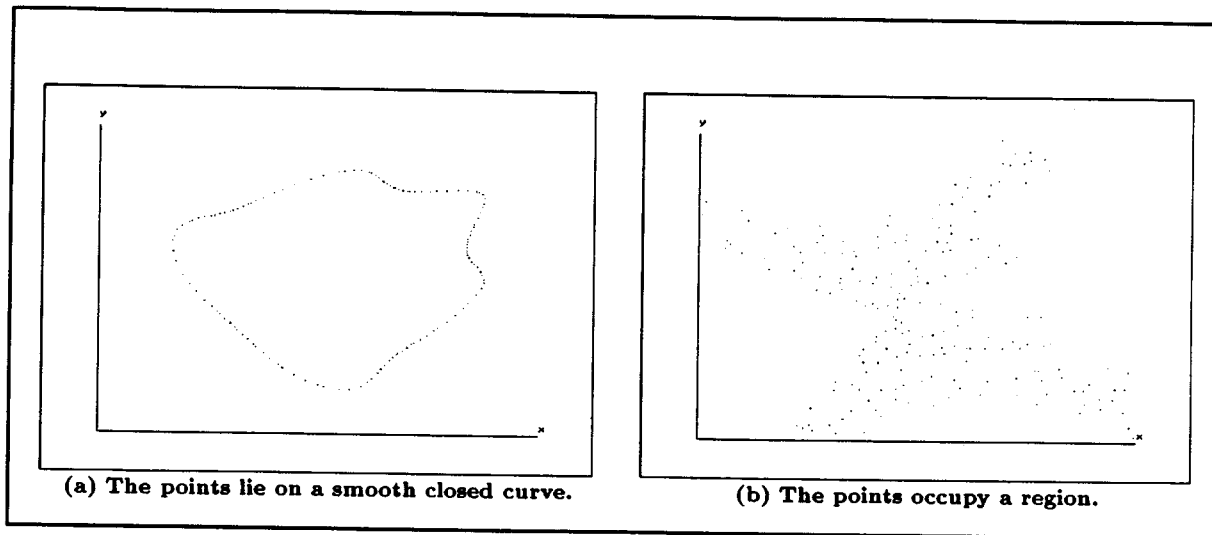


Figure 3.3: *Problem of Topological Dimension: Do the points lie on a curve or cover a region?*

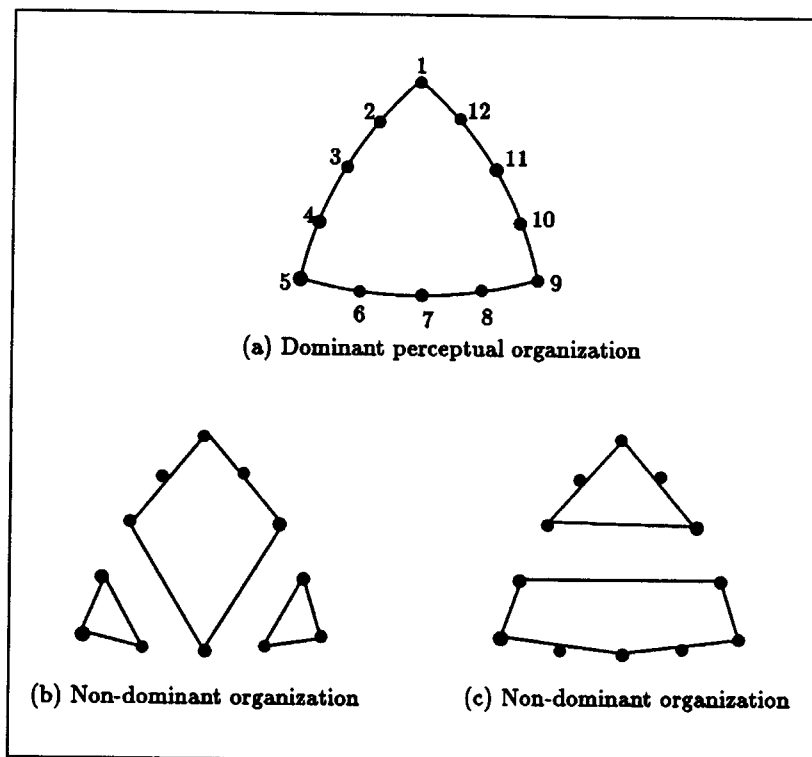


Figure 3.4: *Problem of Sequencing Points: How are the dots ordered?*

of identifying the number of islands in an island chain.

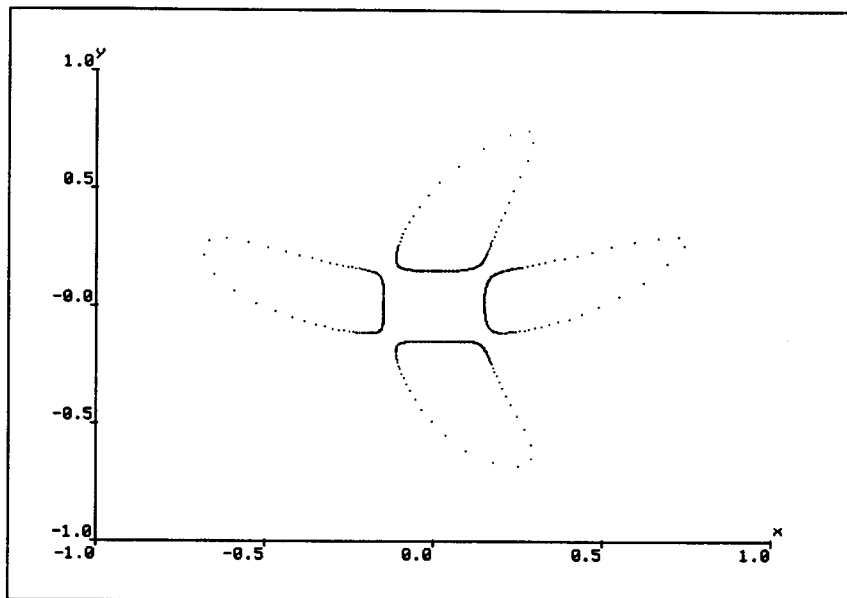


Figure 3.5: *Problem of Clustering - How many clusters are there?*

### 3.1.4 Counting loops of a separatrix

In Fig. 3.6, there is no clear separation between the loops of the separatrix. Yet we would like to be told that there are 5 clusters connected at small “necks”. The number of loops of a separatrix is an important piece of information because it gives the period of the periodic orbit that it encloses.

### 3.1.5 Deciding when steady state is reached

How many iterates are sufficient for revealing the geometric structure of an orbit? Fig. 3.7 illustrates the effects of the number of iterates on what we perceive. The first 192 iterates seem to lie on a single curve. It may lead us to misidentify this dot pattern as a quasiperiodic orbit. Increasing the sample size to 256 iterates, we still perceive a curve. At 320 iterates, we begin to see small branches coming out from the 5 “cusped” portions of the curve. The separatrix does not fully reveal itself until 384 iterates. Frequently,



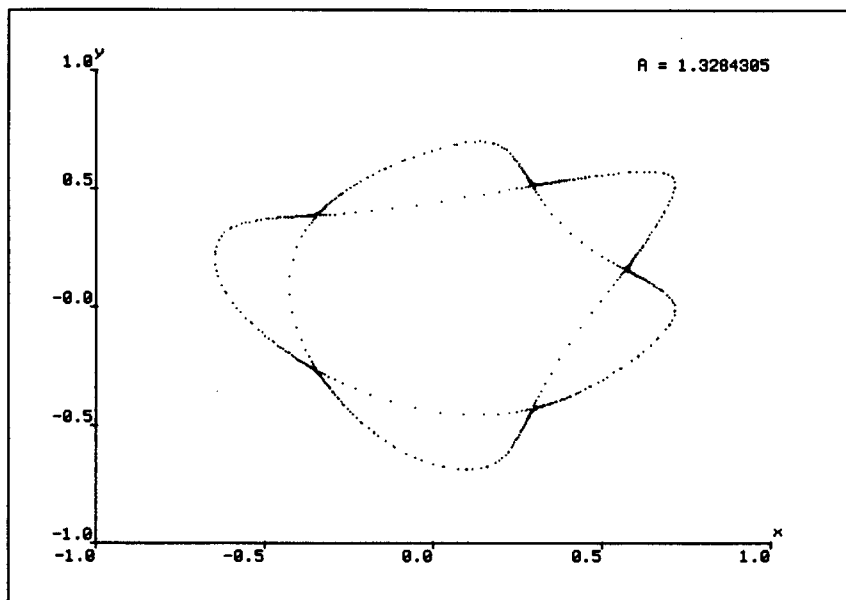


Figure 3.6: *Problem of Loop Counting - How many loops are there?*

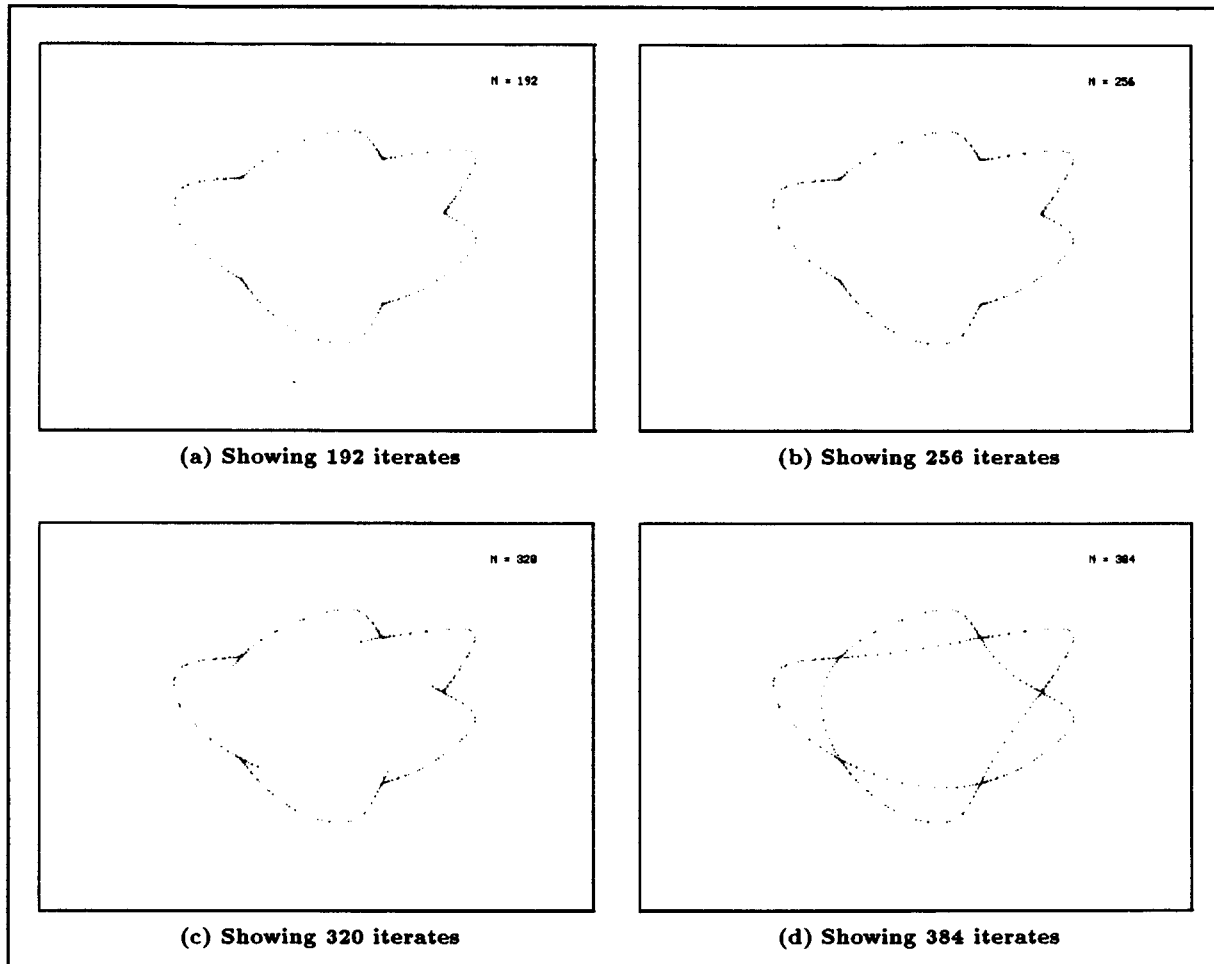
the initial orbit segment of an separatrix can look like a quasiperiodic orbit up to several thousand iterates. So there is a genuine question of how the “steady state” of an orbit is detected.

Another example is provided by the 4-island chain in Fig. 3.8. Looking at the first 128 iterates, we see just 4 disconnected short line segments. In this case, it takes at least 512 iterates for us to clearly perceive 4 distinct closed curves.

### 3.2 CORE IDEA: SHAPE FROM MINIMAL SPANNING TREES

#### What is perceptual grouping?

To perceive a structure in a dot pattern, the perceiver must impose some organization on the dots. In other words, individual isolated dots must be aggregated into larger structures. But given a set of distinct points on the plane, there are many ways to group them. So what are the principles governing dot grouping? When are two dots neighbors of each other? When do these neighboring dots aggregate into lines? Closed curves?



**Figure 3.7: Problem of Steady State: How many iterates does it take to reveal a complete separatrix?**

The objective of a grouping process is to partition the dots into groups such that members within one group are “similar” in some sense, but members belonging to different groups are “dissimilar”. A successful grouping can greatly reduce the combinatorics of the search space when the groups are further organized into higher level structures, and when these structures are matched against idealized models of what we wish to detect.

In this section we will formulate three constraints - soundness, connectivity, and separation – for grouping dots. These constraints are not sufficient to define a mathematical operation, but they will narrow down the class of admissible grouping processes. The goal here is to develop a domain-independent grouping method that gives meaningful structurings of dot patterns and facilitates the subsequent semantic interpretation of an

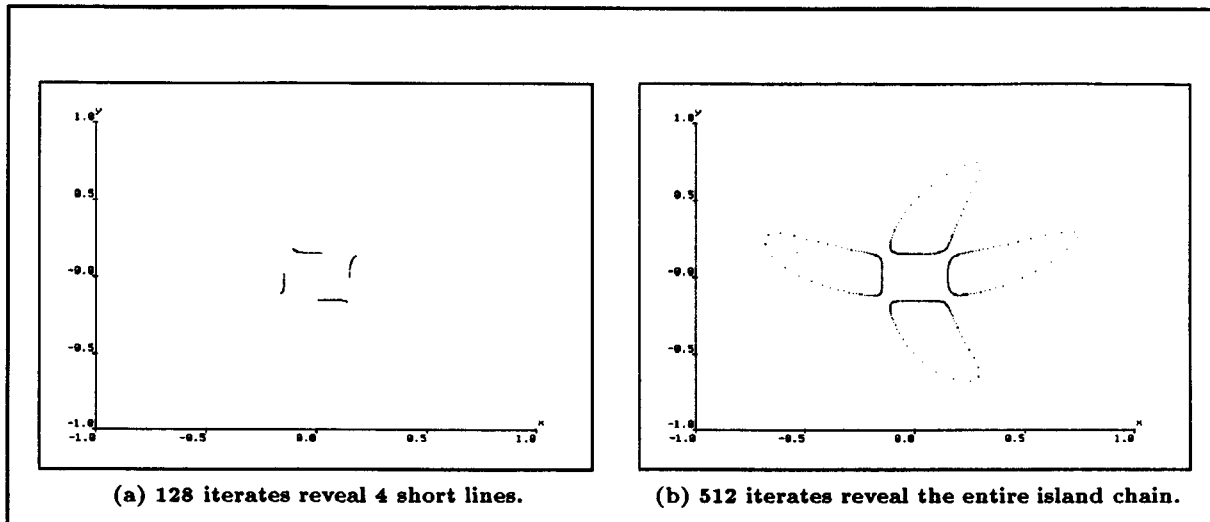


Figure 3.8: *Problem of Steady State: How many iterates does it take to reveal the island structure?*

orbit.

Perhaps the chief contribution of the Gestalt psychologists, who studied human perception of forms in the twenties, is to provide a set of empirical “laws” of organization describing the various factors that influence the tendency for a collection of dots to be grouped together. For example, the *law of proximity* states that dots that are close together tend to be grouped together. The *law of good continuation* suggests that dots are grouped together in such a way to make the fewest separate (non-continuous) contours. The list of organizational laws is quite long, involving laws such as those of *symmetry*, *area*, and *common fate* [12].

The Gestalt laws, however, are at best suggestive, and they remain only ad hoc demonstrations as long as they are not formulated in operational terms. We need, for instance, to be more precise about what we mean by proximity, good continuation, symmetry, etc. We must also consider the combined effects of various grouping factors when several of them are acting in concert or in conflict on any given dot pattern.

As Marr [17] pointed out, the Gestalt movement suffered because the Gestaltists lacked the idea of a *computational process*. They thought of their organization laws as physical laws. But perhaps a more useful perspective is to treat them as *constraints* – properties that a grouping process must satisfy. If enough of these constraints are isolated, one can uniquely characterize a grouping process.

A grouping process groups (or clusters) objects according to the similarity between pairs of objects. Characterizing a grouping process requires the specification of three factors:

- a measurement space in which the objects are described
- a similarity relation between two objects
- a clustering method based on the similarity relation

In the literature on clustering, an object is usually described by a  $m$ -dimensional vector in some measurement (or feature) space, and the proximity between any two objects is defined by some metric relation defined on the measurement space. With dots as the primitive object, the measurement space is extremely simple: each dot is identified by its position in the Euclidean plane <sup>1</sup>.

When do we say two dots are similar? One may say two dots are similar if they are grouped into the same cluster. This definition, however, does not help much because we cannot establish similarity between points until we have computed the grouping, but in order to compute the grouping we need some similarity relations. So either we can come up with a powerful definition of similarity that gives us the grouping right away, or we have to settle on a similarity relation that gives a good approximation to group membership. How good a similarity relation is is measured by the fraction of similar points that are eventually clustered into the same group.

The remainder of this section argues that a geometric structure, the minimal spanning tree, is a basis for defining a good similarity relation.

## Minimal Spanning Trees

Given a set of points in the plane, a **minimal spanning tree** (MST, for short) is a tree of minimum total length whose vertices are the given points. The MST is a useful tool for solving some common network optimizations problems such as the problem of finding the minimum amount of cable required to connect up  $n$  communication centers.

---

<sup>1</sup>We will later consider dots on the surface of a torus, but the essential ideas in both cases are the same.

Another useful feature of a MST is that it gives a definition of when two points are close to each other. Given  $n$  points, we say two points are **adjacent** or **neighbors** if there is an edge connecting them in the MST for the  $n$  points.

Let  $N(p)$  denote the set of all neighbors of a point  $p$ . Then the adjacency relation  $q \in N(p)$  has the following properties:

- It is non-reflexive, i.e.,  $p \notin N(p)$ .
- It is symmetric, i.e.,  $q \in N(p)$  if and only if  $p \in N(q)$ .
- If  $q$  is the nearest point to  $p$  according to the distance metric of the measurement space (i.e.  $\text{distance}(p,q) = \min_{r \in S - \{p\}} \text{distance}(p, r)$ ), then  $q \in N(p)$ .
- It is invariant under similarity transformations (translation, rotation, and uniform scaling of the coordinate axes).

These four properties are also satisfied by three other well-known graph-theoretical definitions of neighbor [24]: the relative neighborhood graph (RNG), Gabriel graph (GG), and Delaunay triangulation (DT). Roughly speaking, these different graphical structures may be thought of as different ways of defining a *region of influence* between two nodes of the graph such that the two nodes are considered neighbors whenever all the remaining nodes lie outside this region of influence. For example, the RNG of a set  $S$  is defined as follows: there is an edge between  $p$  and  $q$  in  $S$  if and only if

$$d(p, q) \leq \min_{r \neq p, q} \max(d(p, r), d(r, q))$$

That means, an edge between  $p$  and  $q$  exists if and only if the region obtained by intersecting the disks of radius length  $(\overline{pq})$  centered at  $p$  and  $q$  respectively contains no point of  $S$  in its interior.

It turns out the MST and these graphs are closely related to each other. Specifically, it can be shown that they form an increasing chain (see [30]):

$$MST \subseteq RNG \subseteq GG \subseteq DT$$

In other words, MST is a subgraph of RNG, RNG is a subgraph of GG, and GG is a subgraph of DT.

At this level of analysis, there is not enough constraint to narrow down our choice of an adjacency definition. However, we will see shortly that the adjacency definition induced

by the MST is consistent with certain natural constraints on the grouping process. To state these constraints, we need some concepts from graph theory.

The points of a dot pattern and their neighbors can be represented by a weighted graph  $G$  – called the **adjacency graph** – consisting of a finite set of nodes and links. The **nodes** of  $G$  are the points of the dot pattern. A **link** (or an **edge**) exists between two distinct nodes if and only if their corresponding points are neighbors. Each link joining two nodes has a value called its **weight**. The numerical value of a link’s weight is taken to be the distance between the corresponding points in the measure space. Graphically, it is convenient to use the length of a link to indicate its weight. A **path** is an alternating sequence of nodes and links such that all the nodes in the sequence are distinct. A graph is **connected** if any two nodes are joined by a path.

One more remark about terminology before we proceed. In what follows, I frequently refer to three different domains: (1) the Geometry World whose objects are points, curves, clusters, and dot pattern, (2) the Dynamics World which provides a semantic interpretation for the geometrical objects, and (3) the Graph World which provides an abstract symbolic representation of the relevant geometrical properties of a dot pattern. Each world has its own descriptive language. To avoid possible confusion, I list the correspondence between terms of the three different languages below:

DYNAMICS	GEOMETRY	GRAPH
iterate	point	node
	neighbor	link (or edge)
orbit	dot pattern	graph (or tree)

### Three constraints

How does the adjacency graph reflect the grouping structures of the dot pattern that it represents? In other words, what are the properties of the adjacency graph that correspond to significant perceptual structures?

If a group of points seems to fall on a smooth curve, then the ordering of points along the curve induces a natural adjacency relation among the points. What does this mean?

Intuitively, we can think of a curve as the path traced out by a particle as it changes its position with time. Using some concepts from topology, we say a **curve** in  $R^2$  is a continuous function mapping the closed unit interval  $[0, 1]$  into  $R^2$ . However, a curve

according to this definition can be quite complicated (see Fig. 3.9). If we are interested in perceptually simple curves, it is necessary to restrict the class of curves we study. In particular, to rule out curves that cross themselves, we define a simple type of curve. We say that a curve  $C$  is a **Jordan curve**<sup>2</sup> if it is topologically equivalent to the closed unit interval  $[0, 1]$  (Fig. 3.9). In other words, there exists a homeomorphism  $f : [0, 1] \rightarrow C$ .  $f$  is often called the **parameterization** of  $C$ .

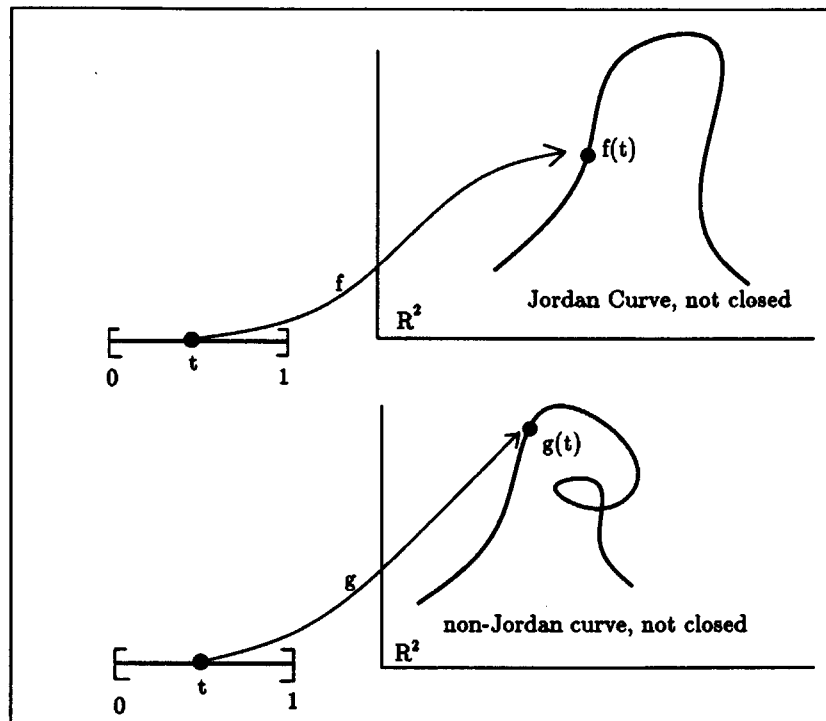


Figure 3.9: A Jordan curve is topologically equivalent to the closed unit interval  $[0, 1]$ .

We discretize the continuum of points in the unit interval by a finite number of isolated points. Since these points are ordered on the real line, we label them – left to right – with integers from 1 to  $m$ . A natural adjacency relation  $N(k)$  can then be defined as follows:

1.  $N(1) = \{ 2 \}$
2.  $N(m) = \{ m-1 \}$

---

<sup>2</sup>In the literature, a Jordan curve is sometimes taken to mean a simple *closed* curve, a curve that is topologically equivalent to the unit circle.

3.  $N(k) = \{ k-1, k+1 \}$ , for  $k \neq 1$  or  $m$ .

What this means is that any isolated point, except the two endpoints, has two neighbors – one on its left and the other right. Via homeomorphism, we can extend this adjacency relation to any Jordan curve. We say a adjacency graph is **sound** if it preserves the natural adjacency relationships for a Jordan curve. This is the content of the **soundness constraint**:

**Constraint 1 (Soundness)** *If a dot pattern  $S$  is perceived as a smooth curve, then its adjacency graph  $G$  should preserve the adjacency relationships implied by a parameterization of the curve.*

Now that we understand what the soundness constraint is, let us see why the constraint is useful. Consider the dot pattern in Fig. 3.10. The dots clearly indicate the letter ‘G’. When we compare the MST and the RNG representation of the dots, we notice an important difference. The MST representation is sound; it captures the dominant perceptual structure of the dot pattern, using a *minimum* number of edges. The RNG representation, on the other hand, is not sound because of the extra edge that “closes” the figure; it does not faithfully represent the topology of the figure ‘G’. The same problem arises in Gabriel graph and Delaunay triangulation since they are supersets of the RNG.

Here is the second constraint, which tries to capture the idea that a cluster should in some sense be “cohesive”:

**Constraint 2 (Connectivity)** *If  $S$  is perceived as a single cluster,  $G$  should be connected.*

The following theorem by Zahn [33] implies that the MST satisfies the connectedness criterion. We first define the distance between two sets  $P$  and  $Q$  of points,  $d(P,Q)$ , as the shortest distance between two points  $p$  and  $q$  such that  $p \in P$  and  $q \in Q$ . Zahn’s theorem is as follows:

**Theorem 1** *Let  $G$  be a graph and  $S$  be the nodes of  $G$ . If  $C$  is a non-empty subset of  $S$  with the property that  $d(P,Q) < d(C,S-C)$  for all partitions  $(P,Q)$  of  $C$ , then the restriction of any MST to the nodes of  $C$  forms a connected subtree of the MST.*



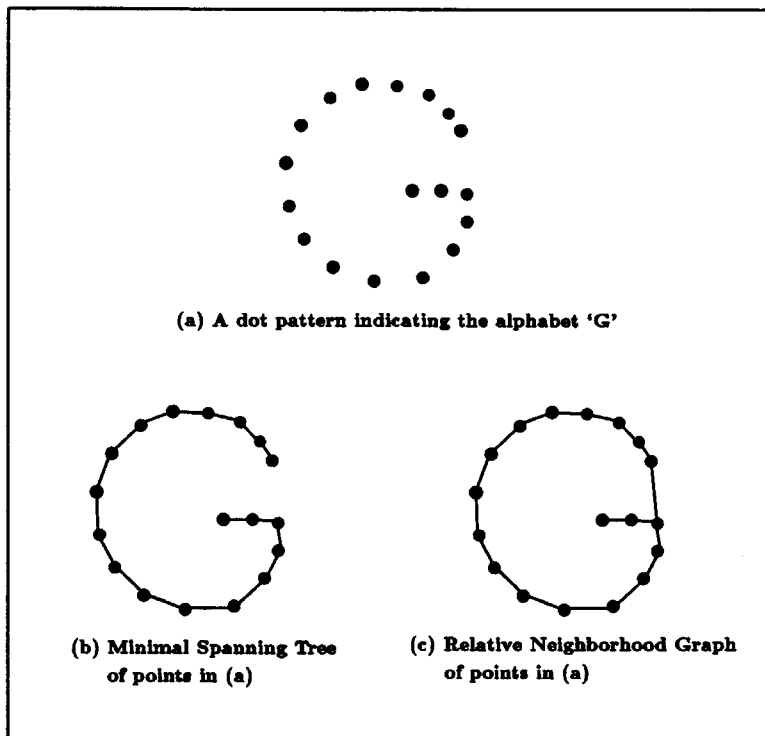


Figure 3.10: *Soundness Constraint. The MST is sound because it captures the structure of the alphabet 'G', but the RNG is not. (Adapted from Fig. 8 of Toussaint.)*

We illustrate the theorem in Fig. 3.11. The dot pattern consists of 17 points. The two clusters,  $C$  and  $S-C$ , indicated by the dotted circles satisfies the hypothesis of theorem 1. This fact can be verified by noting that any partition  $(P, Q)$  of  $C$  is such that  $d(P, Q) \leq 22$ , and  $d(C, S-C) = 78 > 22$ . The theorem implies that the nodes of  $C$  will be a connected subtree. Reversing the role of the partitions  $C$  and  $S-C$ , we can show that the nodes of  $S-C$  also form a connected subtree. So, we can see that the MST has the nice property that it does not break up real clusters. This property, however, is not helpful unless we have a way of detecting which subtrees of the MST do in fact represent real clusters of the dot pattern. For this, we come to our third and final constraint.

The final constraint, the separation constraint, is motivated by the dot patterns in Fig. 3.12. <sup>3</sup>

**Constraint 3 (Separation)** *Suppose  $S$  is perceived as  $n$  distinct clusters,  $C_1, \dots, C_n$ .*

<sup>3</sup>This example is due to Zahn [33].

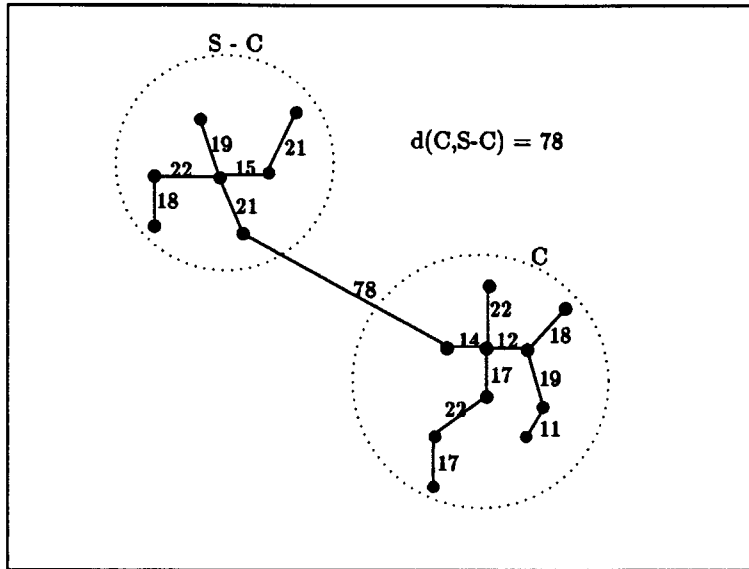


Figure 3.11: *Connectedness Constraint*. The MST representation of a set of 17 points is shown. The subgraphs C and S-C are connected subtrees since they both satisfy theorem 1. (From Zahn 1971))

Let the subgraphs of  $G$  corresponding to these clusters be  $SG_1, \dots, SG_n$ . Then,

1. The nodes of the subgraphs,  $SG_1, \dots, SG_n$ , form a **partition** of the nodes of  $G$ , i.e., the nodes of the  $SG_i$ 's are nonempty, mutually disjoint, and totally exhaustive.
2. The  $SG_i$ 's are either **isolated** from each other (i.e., there is no path connecting  $SG_i$  and  $SG_j$  if  $i \neq j$ ) or **well-separated** (i.e., the links connecting  $SG_i$  and  $SG_j$ ,  $i \neq j$ , are significantly longer than the nearby links within  $SG_i$  or  $SG_j$ ).

The two clusters in Fig. 3.12a-1 are **homogeneous** in the sense that the distance between a point to its nearest neighbor is approximately the same for every point in a cluster. These two clusters are also **well-separated** because the inter-cluster distance  $D$  is much larger than the average within-cluster nearest-neighbor distance  $d$ . This observation gives a simple clustering method: connect any pair of points whose distance is less than a threshold value depending on  $d$ . For instance, the graph in Fig. 3.12a-2 is obtained by using a threshold of  $d + 2\sigma$ , where  $\sigma$  is the standard deviation of the set of nearest-neighbor distances. As we can see, the graph has two *isolated* connected subgraphs corresponding to the two clusters that we intuitively perceive.

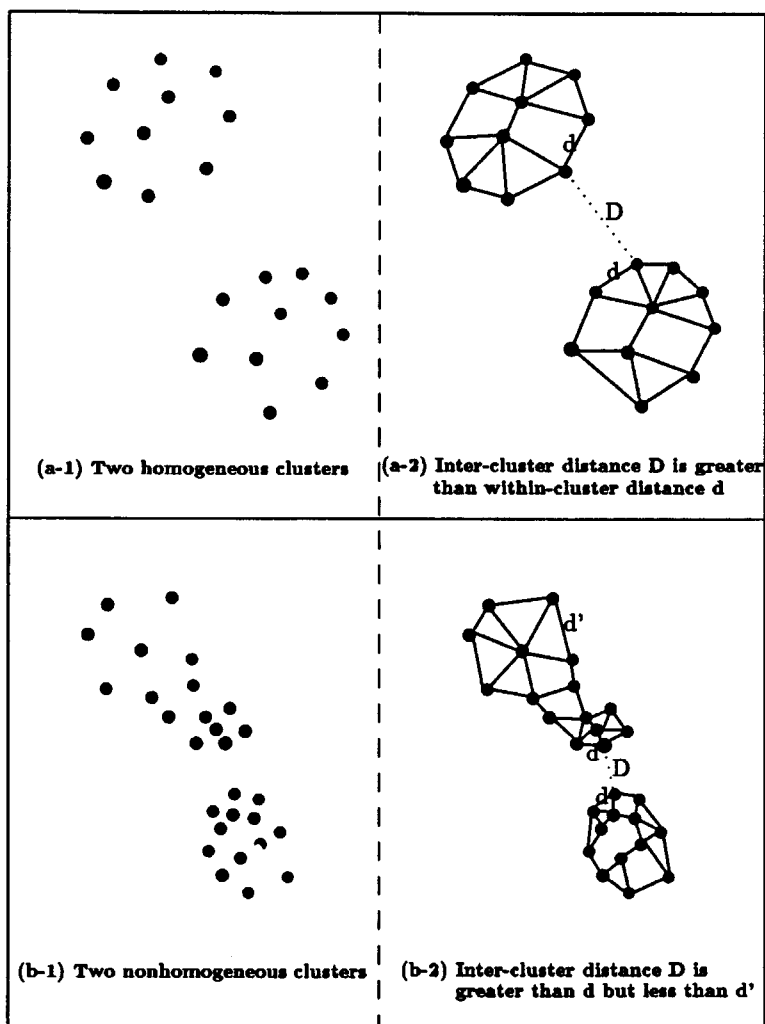


Figure 3.12: *Separation Constraint: Two clusters are well-separated if the inter-cluster distance  $D$  is significantly larger than the nearby within-cluster distances  $d$ .*

The simple method of comparing inter-cluster with within-cluster distances, however, does not quite work for nonhomogeneous clusters such as those in Fig. 3.12b-1. Here, although the inter-cluster distance  $D$  is greater than the nearby within-cluster distance  $d$ , it is *less than* some within-cluster distance  $d'$  of the upper cluster (Fig. 3.12b-2). We can resuscitate the simple method by restricting the distance comparison to a local region around the inter-cluster edge, i.e., the statistics  $d$  and  $\sigma$  are taken locally. As an example, the graph in Fig. 3.12b-2 is generated by computing the edge length statistics based on the 3-nearest neighbors of each point. Once again, we have two isolated connected subgraphs

because the inter-cluster edge is significantly longer than its nearby edges.

## Upshot

This section defined an adjacency relation between two points based on MST. The adjacency relation is non-reflexive, symmetric, non-empty, and invariant under the similarity transformations. I also formulate three constraints – soundness, connectedness, and separation – that are motivated by the desire to have a faithful graphical representation of the grouping structure of a dot pattern. Finally I show that the proposed adjacency relation satisfies all three constraints. In the next section, we will see how shape information are extracted from the MST representation.

### 3.3 SHAPE FROM MINIMAL SPANNING TREE

As discussed in section 3.1, in order to direct the course of its numerical experiments, KAM must be able to interpret what it sees. In particular, KAM must be able to recognize the relevant geometrical properties of a collection of iterates. Orbit recognition consists of the following five tasks:

1. to tell when the iterates seem to lie on a curve
2. to detect clusters
3. to count the number of loops of a separatrix
4. to identify a chaotic orbit
5. to determine when an orbit has reached a “steady state”

This section shows how these tasks can be accomplished by examining the MST representation of a set of iterates. To begin, we need some terminologies from graph theory.

A cycle is a path whose start and end nodes are the same. A tree is a connected graph with no cycles. A spanning tree of a connected graph  $G$  is a tree that contains all the nodes of  $G$ . The weight of a tree is the sum of the lengths of its links. A minimal spanning tree is a spanning tree whose weight is minimum among all the spanning trees

of  $G$ . The **degree** of a node  $p$ ,  $\text{deg } p$ , of a minimal spanning tree  $T$  is the number of links incident with  $p$ . A **branch** at a node  $p$  of  $T$  is a maximal <sup>4</sup> subtree containing  $p$  as an endpoint. Thus, the number of branches at  $p$  is also equal to  $\text{deg } p$ . A node  $p$  is a **branching node** if  $\text{deg } p > 2$ . A node  $p$  is a **terminal node** if  $\text{deg } p = 1$ . The distance  $d(p,q)$  between two nodes  $p$  and  $q$  of  $T$  is the length of the shortest path joining them. A shortest  $p$ - $q$  path is often called a **geodesic**. A diameter  $\text{diam}(T)$  of a tree  $T$  is any longest geodesic.

### 3.3.1 Determining when iterates appear as a closed curve

Let  $S$  be a set of iterates  $(x_1, y_1), \dots, (x_n, y_n)$ , and  $\text{MST}(S)$  be the minimal spanning tree of  $S$ . If  $\text{MST}(S)$  has no branching nodes, then we may generate a plane curve interpolating  $S$  in terms of a parameter  $u$  as follows:

$$\begin{aligned}x &= x(u) \\ y &= y(u)\end{aligned}$$

A natural choice for the parameter  $u$  is the path length with respect to a terminal node of  $\text{MST}(S)$ . Since a tree has no cycles, the parametric curve so generated from  $\text{MST}(S)$  is a Jordan curve. We thus conclude that if the MST representing the initial segment of an orbit has no node with degree greater than two, i.e., no branching node, then there exists a Jordan curve interpolating the iterates of the orbit segment.

To identify a quasiperiodic orbit, we need a stronger conclusion, namely, whether the Jordan curve is closed. But this is simple. Since  $\text{MST}(S)$  for a Jordan curve has only two terminal nodes, all we need to do is to note the distance between these two terminal nodes. If, in addition to having no branching node,  $\text{MST}(S)$  has its two terminal nodes close to each other on the Euclidean plane, then we conclude  $S$  lies on a closed Jordan curve.

There is an extra bonus if  $\text{MST}(S)$  generates a parametric curve. The parameter  $u$ , which is taken to be the path length, gives an *ordered* sequence of the iterates. This sequencing is going to be useful for extracting further shape information from the curve such as the locations of extreme curvature along the curve.

---

<sup>4</sup>A graph is **maximal** if it is not contained in any larger graph of the same sort.

### 3.3.2 Partitioning iterates into distinct clusters

The iterates of an island chain fall into distinct islands. KAM must be able to determine the number of islands because this number gives the period of the enclosing periodic orbit. The question is how to detect clustering structure in the minimal spanning tree.

A MST, by definition, is a connected graph; it does not force “breaks” in the representation even when there are real gaps among the points in the dot pattern. However, as we have seen the MST edges within a cluster tend to be short, while edges joining clusters are in general much longer than nearby within-cluster edges. If these longer intercluster edges can be detected, then we have a way to partition the dot pattern because deleting these long edges breaks up the MST into several subtrees each of which corresponds to a cluster.

Following a suggestion by Zahn [33], we call a MST edge **inconsistent** if it is significantly longer than nearby edges. There are several ways to define an inconsistent edge. One reasonable criterion is to compare an edge’s length  $W$  with the average  $m$  and standard deviation  $\sigma$  of nearby edges. For example, the following definition is used in KAM:

**Definition 1** *An edge with length  $W$  is inconsistent if  $W > m + 2\sigma$ .*

In other words, an edge is inconsistent if its length exceeds the average of the nearby edge lengths by two standard deviations.

One question remains: Which are the nearby edges? Let  $e$  be an edge joining nodes  $p$  and  $q$ . The idea is that nearby edges are those that are within certain distance from  $e$ . The following criterion is chosen empirically, and seems to give correct results:

**Definition 2** *The nearby edges of  $e$  are edges that lie on a path through either  $p$  or  $q$  containing 3 or fewer edges.*

### 3.3.3 Counting the number of loops of a separatrix

Partitioning the points of a separatrix is a difficult clustering problem because the clusters are not well-separated. According to the clustering method of deleting inconsistent edges,

a separatrix, such as the one shown in Fig. 3.13a, will appear as a single cluster. But we can see the separatrix as consisting of five loops connecting at the small “necks” around the hyperbolic points (see the boxed region in Fig. 3.13a). It turns out the geometry of the points around these necks gives us a clue to solving this clustering problem.

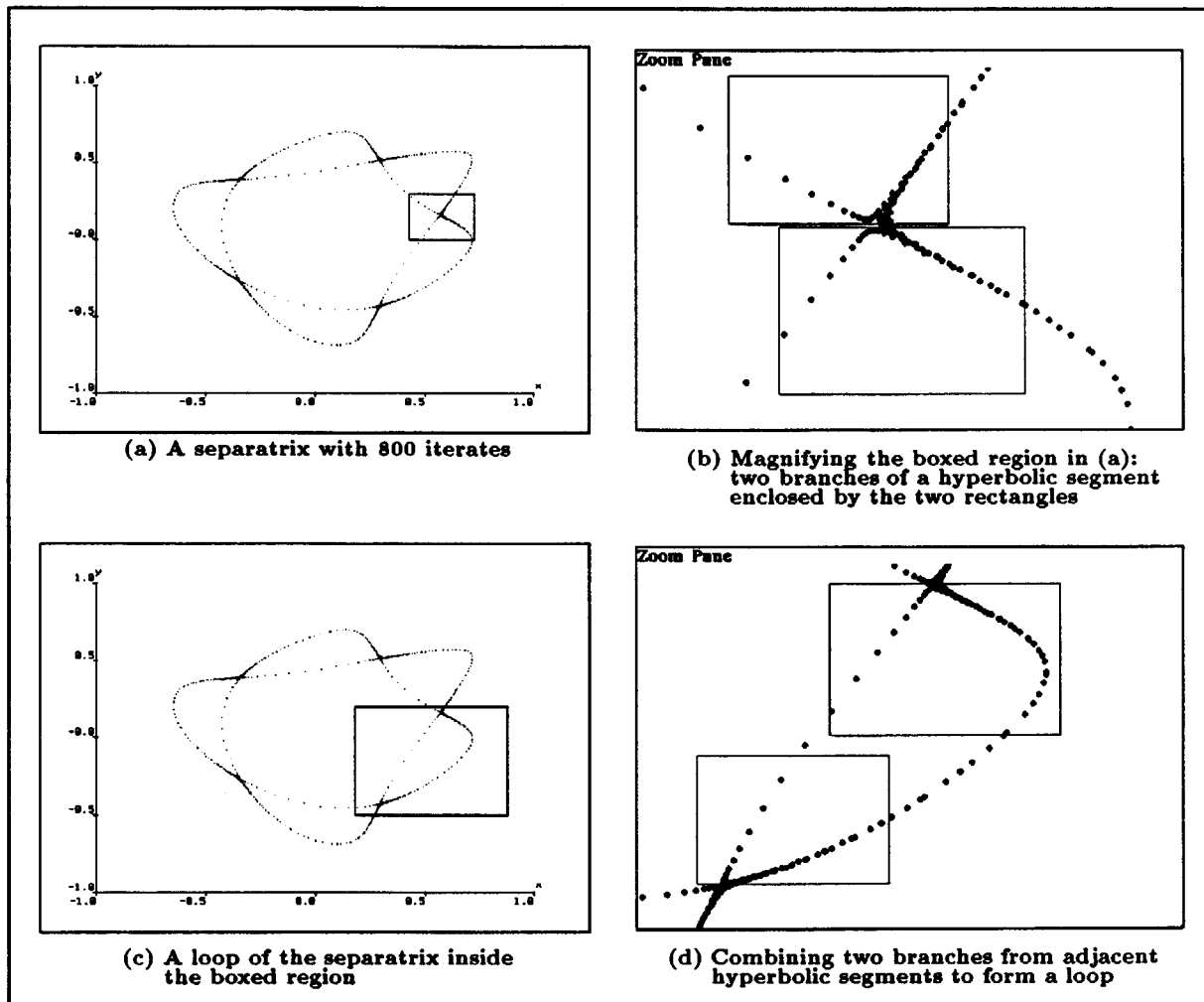


Figure 3.13: *Hyperbolic segments of a separatrix*

Notice that the points on the two sides of a neck have a distinctive geometry, namely, they seem to lie on the two branches of a hyperbola. Let me call this arrangement a **hyperbolic segment** (Fig. 3.13b). The 5-loop separatrix can then be thought of as consisting of 5 such hyperbolic segments. So if we can detect a hyperbolic segment, we will have a method to partition the points. Specifically, for two adjacent hyperbolic segments, we group together points that lie on the two nearby hyperbolic branches, one from each of the hyperbolic segments (Fig. 3.13c,d).

But how is a hyperbolic segment recognized? We will see once again the MST representation contains sufficient information to answer this question. Before explaining the recognition method, let me recall the definition of a diameter of a tree: A diameter is any longest geodesic. If the MST has a “linear structure”, i.e., it has no branching node, then the diameter is unique and every point belongs to it. For a dot pattern covering a region, its corresponding MST in general has many branching nodes along a diameter. 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. 3.13b).

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 zeroes. 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 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. 3.14a 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. 3.14b. 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 4 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.

### 3.3.4 Identifying Chaotic Orbits

In the previous three sections, we have seen how the clustering and branching structure of the MST representation of a dot pattern allow us to recognize a quasiperiodic orbit, an island chain, and a separatrix. The purpose of this section is to show how a chaotic orbit is positively identified by an additional geometric property that can be extracted from its MST representation – the percentage of **primary nodes**.

We'll begin with some definitions. Suppose  $G$  is a MST and  $\text{diam}(G)$  is its diameter. A **diameter node** of  $G$  is a node that lies on  $\text{diam}(G)$ . A **diameter branch node** is a branching node that lies on  $\text{diam}(G)$ . It is indicated by the BRANCH symbol in the symbolic description of the branching structure. A **primary branch** is any longest branch rooted at a diameter branch node. A **primary branch node** is a node that belongs to some primary branch. Finally, a **primary node** is either a diameter node or a primary branching node.

The idea of a primary node is motivated by the rich branching of a MST representing a chaotic orbit. Compare the branching structure of the three different types of orbits <sup>5</sup> in Fig. 3.15. All the nodes of an island chain are either diameter nodes or nodes that lie on the main branches coming out of the diameter. For a separatrix, almost all nodes are primary with the exception of a few nodes usually located near the hyperbolic points. Incidentally, the non-primary nodes are too small to be seen in Fig. 3.15b-2. And this

---

<sup>5</sup>I have not included a quasiperiodic orbit in the picture because it is clear that all its nodes are diameter nodes and hence are primary.

is precisely the point. For if we contrast Fig. 3.15b-2 with Fig. 3.15c-2, we notice the prominence and abundance of non-primary nodes (nodes that are not covered by black dots or open circles) for the chaotic orbit. In the particular chaotic orbit shown, roughly 40% of all the nodes are non-primary.

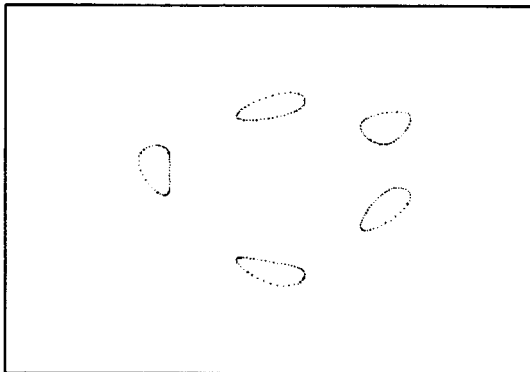
With primary nodes we have a simple rule for recognizing a chaotic orbit: If the percentage of the primary nodes of a MST is less than certain threshold (say, 80%), then we conclude that the MST represents a chaotic orbit. Note that all the nodes of a quasiperiodic orbit or an island chain are primary. For a separatrix, most of its nodes are primary; the few non-primary nodes are usually located in a small region around the hyperbolic points.

### 3.3.5 Determining steady state

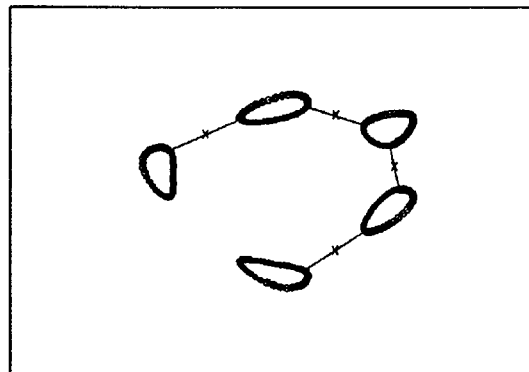
The problem of steady state is the problem of deciding how many iterates are needed for identifying an orbit. There is a tradeoff involved. In the limit of using a very large number of iterates, a reliable classification is achieved because the probability of missing part of the orbit structure goes 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 straightforward 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. 3.14b) . 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.

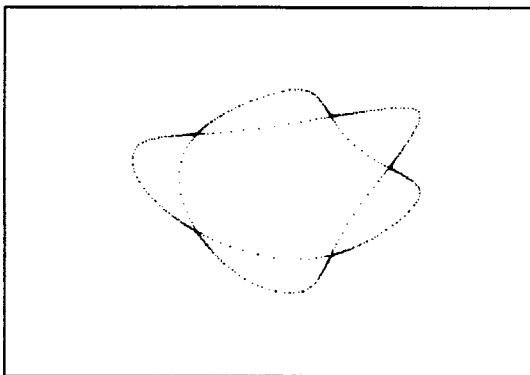
Of course, there must be a lower bound on the number of iterates to be added at each stage. If we add just a few iterates to a large set of iterates, we certainly do not expect any change in the branching structure. In my experiments, I set the lower bound to a fixed percentage (say 25%) of the current number of iterates in the set. Suppose, for instance, there are 1000 iterates in the set. Then at least 250 new iterates should be added in the next stage. This technique turns out to work quite well as we will see later.



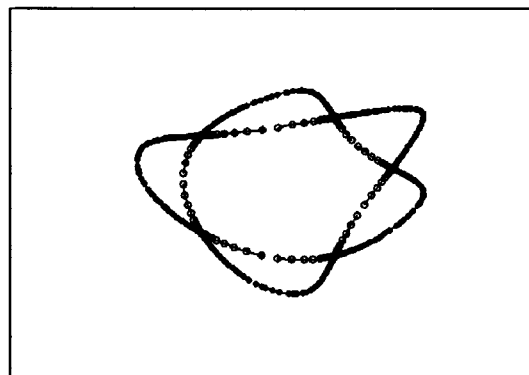
(a-1) A 5-island chain with 384 iterates.



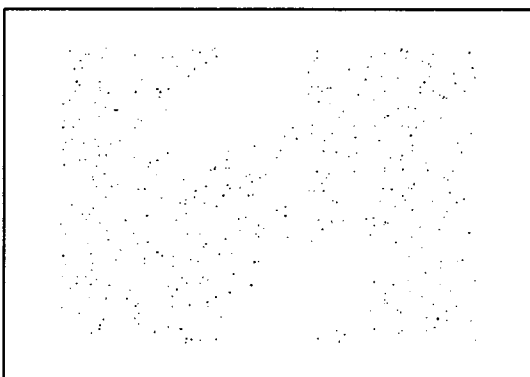
(a-2) Number of non-primary nodes = 0.  
Percentage of primary nodes = 100%.



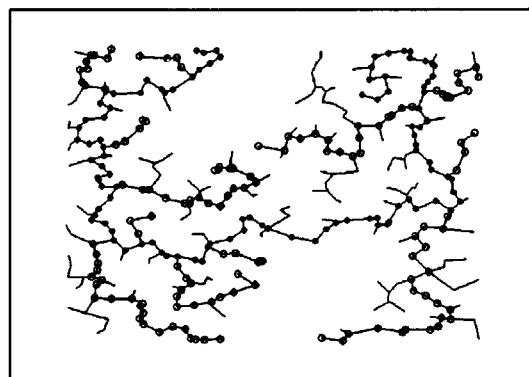
(b-1) A separatrix with 800 iterates.



(b-2) Number of non-primary nodes = 65.  
Percentage of primary nodes = 92%.



(c-1) A chaotic orbit with 384 iterates.



(c-2) Number of non-primary nodes = 150.  
Percentage of primary nodes = 61%.

Figure 3.15: A chaotic orbit can be distinguished from other orbits by the smaller percentage of primary nodes. Primary nodes are indicated by black dots ( $\bullet$  = diameter nodes) or open circles ( $\circ$  = primary branch nodes).

## 3.4 IMPLEMENTATION

### 3.4.1 Computing the Minimal Spanning Tree

The minimal spanning tree (MST) comes up frequently in network optimization problems. For example, if one desires to build a communication system among  $n$  cities requiring interconnecting cables, then the MST results in a network of minimum cost (assuming cost is proportional to the length of the cable used).

Unlike many other path minimization problems – such as the traveling salesman problem, the Steiner tree problem – the MST can be solved by an efficient algorithm. One such algorithm was devised by Prim [25]. The idea is quite simple: one starts with any arbitrary node and then branches out from it by choosing edges according to a certain strategy. Specifically, one partitions the nodes of the graph into two disjoint sets,  $U$  and  $V$ . Initially,  $U$  contains only an arbitrary node  $p_1$ , and  $V$  the remaining nodes. The next step is to choose a node  $p_2$  from  $V$  such that the distance between  $p_1$  and  $p_2$  is the minimum among all edges with one endpoint in  $U$  and the other endpoint in  $V$ . Enlarge  $U$  by joining  $p_2$  to  $p_1$ , and remove  $p_2$  from  $V$ . One iteratively enlarges  $U$ , each time choosing a node from  $V$  that is closest to the set  $U$ . The algorithm terminates when  $U$  contains all the nodes. It can be easily shown that Prim's algorithm takes  $O(n^2)$  time where  $n$  is the number of nodes.

Prim's algorithm is optimal for a general weighted graph, a graph whose edge weights are unrestricted, but is suboptimal if the graph has a geometric structure. Specifically, Shamos [24] has shown that if one considers points in the Euclidean plane, the MST can be constructed in  $O(n \log n)$  time. This result should not be surprising because the metric properties of the Euclidean plane give more constraints to the Euclidean MST problem. Unfortunately, the time complexity is an asymptotic result. Almost all the known  $O(n \log n)$  methods require the construction of a Voronoi Diagram of the given point set as an intermediate step. My own experimentation with these optimal methods indicates the crossover point is well over several thousand points. So, for the purpose of orbit identification, I still rely on the simple Prim's algorithm.

### 3.4.2 Computing the diameter of MST

Recall a diameter of a MST is any longest geodesic (section 3.3). A brute-force approach to compute a diameter – for example, by finding all possible geodesics and selecting the

longest one – would be extremely inefficient. But the definition of diameter implies that its endpoints must be terminal nodes (otherwise, we can extend the geodesic to obtain a longer geodesic, thus violating that it is a diameter). However, even computing the geodesics between all possible pairs of MST terminal nodes can be quite expensive if the MST contains many terminal nodes.

Here we need to exploit some domain knowledge to cut down the cost of computation. For regular orbits, like quasiperiodic orbits or island chains, the number of terminal nodes is small. So we can afford to compute all the geodesics between terminal nodes. For chaotic orbits, there are lots of terminal nodes. But in this case it does not really matter whether we can find the longest geodesic. Any reasonably long geodesic will contain sufficient branching structure for identifying the orbit type. We will say a geodesic between two terminal nodes is a **near diameter** if there is no longer geodesic originating from either one of the two nodes.

It is straightforward to compute a near diameter. Pick any two terminal nodes. Compute its geodesic. Walk along the geodesic to check if there are any branches coming out from the geodesic that will generate a longer path. If so, pick this longer path, and repeat the above steps. Otherwise, report that the geodesic is a near diameter.

### 3.4.3 Orbit Classification Rules

We have seen the main ideas behind obtaining the shape of an orbit from its MST representation in section 3.3. This knowledge is encoded in KAM by 5 orbit classification rules which are collected in Fig. 3.16. Since the rules are largely self-explanatory, I will just make two clarifying remarks. The first remark concerns the absence of a “Periodic Orbit Identification Rule”. The problem with periodic orbits is that we almost never directly observe them in numerical experiments because of round-off errors. Even if we start the initial state precisely at a periodic point, the trajectory will drift to a nearby quasiperiodic orbit after several iterations. So, we do not really need to directly recognize a periodic orbit; rather, its existence and location are inferred from the smallest enclosing quasiperiodic orbit.

The second remark has to do with the Escape Orbit Identification Rule. The question is how to decide an orbit is “escaping”. KAM is not very smart about this. In the beginning of an experiment session, KAM asks the user to supply an **escape region**. The escape region is usually given as the exterior of a closed and bounded set in the phase space. An iterate is **out of range** when it is inside the escape region.

## **Orbit Identification Rules**

### **Quasiperiodic Orbit Identification Rule**

If (1)  $MST_n$  has only one cluster,  
(2)  $MST_n$  has no branching node,  
(3) The distance between the initial and final point of the diameter of  $MST_n$  is less than 5% of the total diameter length,  
then the orbit must be a QUASIPERIODIC ORBIT.

### **Island Chain Identification Rule**

If (1)  $MST_n$  has  $k (> 1)$  clusters,  
(2) Each cluster of  $MST_n$  is a quasiperiodic orbit,  
then the orbit must be a  $k$ -ISLAND-CHAIN.

### **Separatrix Identification Rule**

If (1)  $MST_n$  has a hyperbolic structure,  
(2)  $MST_n$  has  $k (> 1)$  pairs of hyperbolic branches,  
(3) Over 80% of the  $MST_n$  nodes are primary,  
then the orbit must be a  $k$ -SEPARATRIX.

### **Chaotic Orbit Identification Rule**

If less than 80% of the  $MST_n$  nodes are primary,  
then the orbit must be a CHAOTIC ORBIT.

### **Escape Orbit Identification Rule**

If any iterate is out of range,  
then the orbit must be an ESCAPE ORBIT.

Figure 3.16: Five orbit classification rules.  $MST_n$  is the minimal spanning tree constructed at stage  $n$ .

### **3.4.4 Orbit Recognition Algorithm**

The purpose of the Orbit Recognition Algorithm is to classify a finite orbit segment into one of the known orbit types. The algorithm takes four inputs: (1) an area-preserving

map, (2) an initial state, (3) a parameter value, and (4) an escape region. The output is typically a known orbit type such as QUASIPERIODIC, ISLAND-CHAIN, etc. However, the algorithm can give up, in the sense that up to a certain threshold number of iterates it still cannot reliably identify the type of the orbit segment. When the recognition algorithm gives up, KAM will dispatch to the Phase Space Searching Module, asking for a new initial state.

The recognition routine is organized into two nested levels. The lower level is the piece I call the **identification routine**. This routine implements the **aggregate-partition-classify** algorithm. It has three steps:

- It first aggregates a set of iterates into a MST, thus creating the adjacency graph. Relevant geometrical properties of the MST such as its diameter and the branching histogram are also computed.
- It then partitions the adjacency graph into clusters by detecting and removing inconsistent edges of the MST.
- It finally runs the orbit identification rules, trying to detect line structure, hyperbolic structure, or line structure within a cluster component.

The identification routine does not make any decision about whether the identification result is stable or robust, i.e, whether aggregating a larger set of iterates will produce the same identification. That decision is encoded in the higher level **stability-control routine**. The purpose of the stability control is to achieve some robustness in the classification result with minimum computational effort. The fundamental task of the stability-control routine is to arrive at the smallest number of iterates that yield a stable classification. The routine implements a straightforward technique: *iterative perturbation*. The idea is we start with some initial set  $S_0$  consisting of a small number of iterates (say,  $N_0 = 128$ ), and *perturb*  $S_0$  by adding  $\Delta N_0$  iterates. Call the larger set of iterates  $S_1$ . We can iterate this perturbation process to produce a sequence of larger sets,  $S_0, S_1, S_2, \dots, S_m$ . The stability-control routine calls the identification routine with each  $S_i$  successively. We terminate the iterative process when the last  $k$  members of the sequence all yield the same orbit class.

In the current implementation,  $k$  is chosen to be 2 and there is an upper limit ( $m = 7$ ) on the number of iterations. The upper limit is set because the time it takes to compute the MST of a set of size  $n$  grows as  $O(n^2)$ . If a faster implementation of the MST



algorithm is available, we can increase  $m$  accordingly. The number of iterates added at each iteration is somewhat arbitrarily chosen as follows. Let  $N_i$  be the number of iterates in  $S_i$ , and  $\Delta N_i$  be the number of iterates to be added at stage  $i$ . Then, we have:

$$\begin{aligned} N_0 &= 128 \\ N_{i+1} &= N_i + \Delta N_i \\ \Delta N_i &= \max(128, \frac{1}{4} \times N_i) \end{aligned}$$

The explicit value of each  $N_i$  is listed below:

$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$
128	256	384	512	640	800	1000	1250

### 3.5 EXPERIMENT: MAIN ILLUSTRATION

To illustrate the orbit recognition algorithm, I give an example of how KAM recognizes a 4-island chain generated by the Henon map with parameter value  $\alpha = 1.5808$  (i.e.,  $\cos \alpha = -0.01$ ) and initial condition  $(0.15, 0)$ . The total running time for this experiment is approximately 9 minutes on a Symbolics. KAM requires 640 iterates to classify this orbit. Note that the number of iterates required is not far from the optimal number since, as we have seen in section 3.1, at least 512 iterates are needed to reveal the entire structure of this particular island chain. Let us run through the experiment. Five snapshots of the experiment are shown in the following five pages (Fig. 3.17).

The first snapshot shows the result of trying 128 iterates. The dot pattern consists of 4 isolated short line segments. The clustering algorithm correctly finds the 3 inconsistent edges and identifies the 4 clusters. Each cluster, however, does not represent a quasiperiodic orbit. So, the recognition algorithm does not assign any category to this set of iterates. It goes on to request for more iterates.

The second trial examines 256 iterates. The dot pattern changes. The short line segments have grown into longer curves, but these curves are not yet closed because of the large gaps between the endpoints. The clustering algorithm again correctly identifies the 4 clusters. Just as the previous case, the recognition algorithm does not classify the iterates into any orbit type.



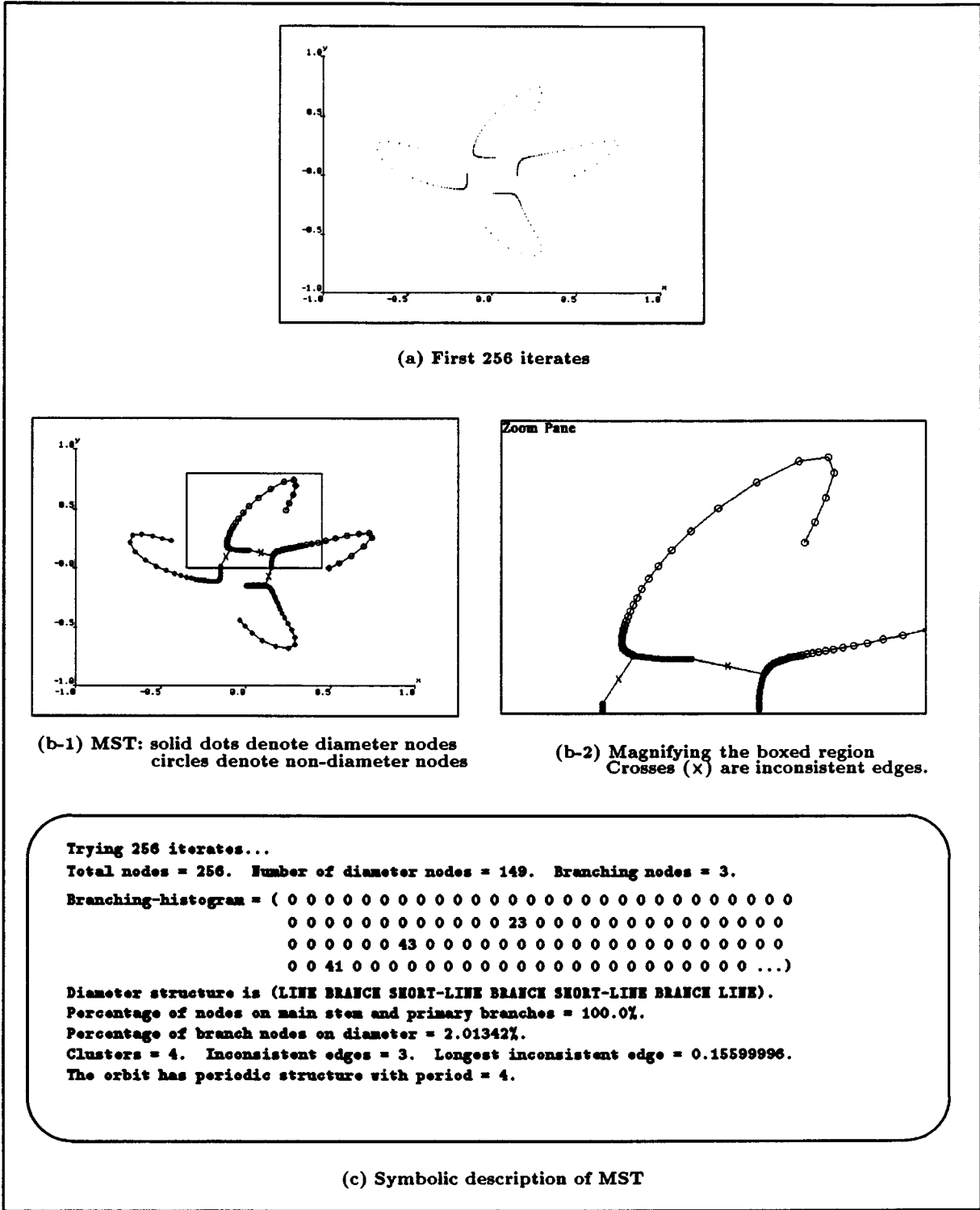
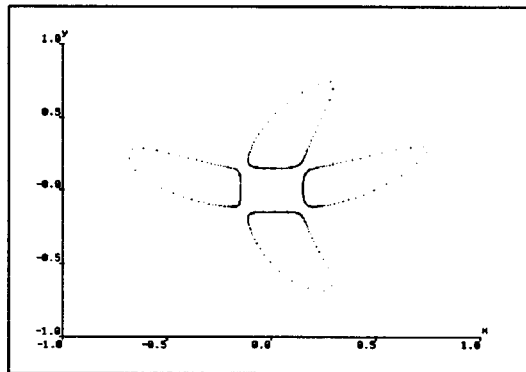
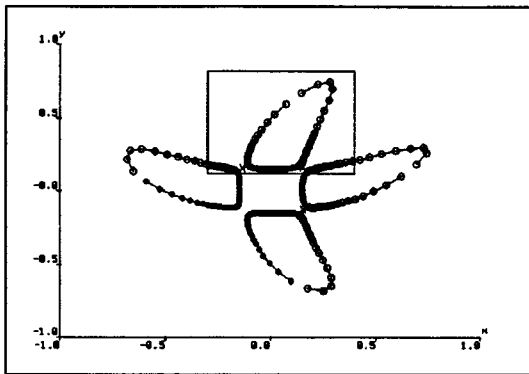


Figure 3.17: (cont.) Recognizing a 4-island chain from the first 256 iterates.

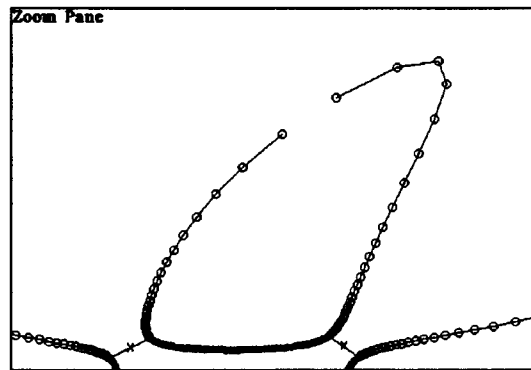




(a) First 512 iterates



(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 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%.

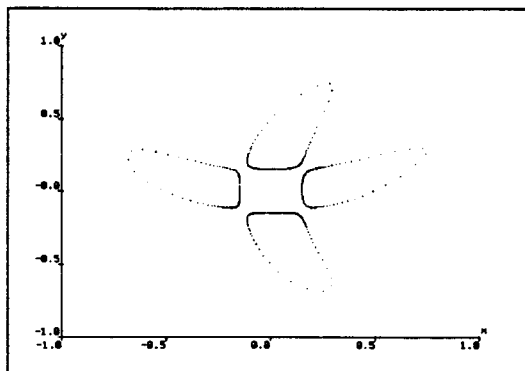
Percentage of branch nodes on diameter = 1.7699127%.

Clusters = 4. Inconsistent edges = 3. Longest inconsistent edge = 0.06967917.

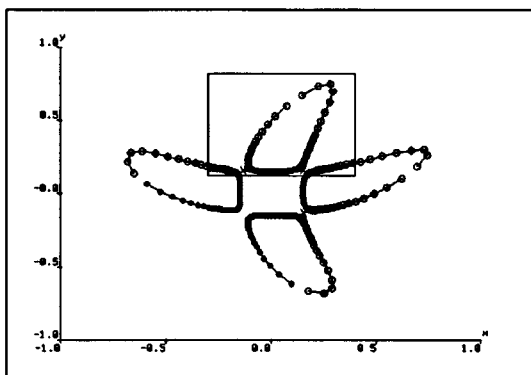
The orbit has periodic structure with period = 4.

(c) Symbolic description of MST

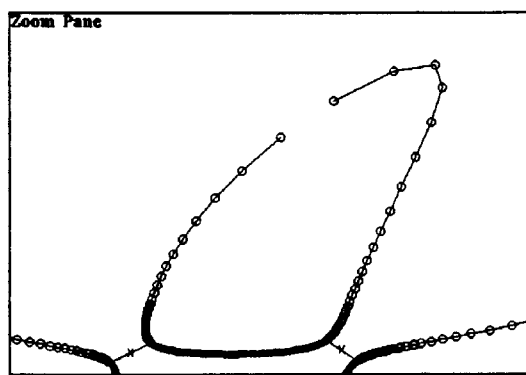
Figure 3.17: (cont.) Recognizing a 4-island chain from the first 512 iterates.



(a) First 640 iterates



(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 640 iterates...
Total nodes = 640. Number of diameter nodes = 419. Branching nodes = 6.
Branching-histogram = (0 ... 0 30 43 0 ... 0 30 45 0 ... 0 27 46 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 = 1.431984%.
Clusters = 4. Inconsistent edges = 3. Longest inconsistent edge = 0.06960424.
The orbit has periodic structure with period = 4.
Process component orbit... Looking at every 4 iterates...
The component orbit reaches steady state at 128 iterates.
The orbit reaches steady state at 640 iterates.
The iterates form a QUASIPERIODIC ISLAND CHAIN with 4 islands.

```

(c) Symbolic description of MST

Figure 3.17: (cont.) Recognizing a 4-island chain from the first 640 iterates.

The result of increasing to 384 iterates is shown in the next snapshot. The 4 open curves have grown even longer, and the gap between the endpoints narrowed. But the gaps are still too large for the recognition algorithm to identify them as closed curves. Unlike the previous two trials, the clustering algorithm finds 7 inconsistent edges and hence 8 clusters. These clusters correspond to the left and right halves of the islands. Again, no orbit category is assigned.

Next, the number of iterates is increased to 512. We clearly perceive 4 isolated closed curves in the dot pattern. The clustering algorithm correctly identifies the 4 clusters. The recognition then randomly selects one of these islands, and verifies that it represents a quasiperiodic orbit. Notice also the symbolic description of the branching structure: three pairs of BRANCH symbols appear. Each BRANCH symbol represents a long primary branch rooted at the corresponding diameter branch node. These primary branches are the left and right halves of the islands.

The fifth and final snapshot contains the result of identifying 640 iterates. The clustering and branching structures of the MST are the same as the previous case. The recognition algorithm randomly chooses one of the 4 islands, and tests if it is a quasiperiodic orbit. The test result is positive. So, the category of QUASIPERIODIC ISLAND CHAIN is assigned to this orbit segment consisting of 640 iterates. Since both the current and previous trials result in the same orbit type, the recognition algorithm announces that steady state has been reached, and reports its classification result.

### **3.6 ANOTHER EXPERIMENT: ORBIT ON SURFACE OF A TORUS**

The orbit recognition algorithm has been successfully tested with thousands of orbits generated by the Henon map, Bak Map [4], and Standard Map [14]. This section will show an experiment to identify a 2-loop separatrix of the Standard Map with parameter value  $K = 0.6$  and initial condition = (0.77310216, 0.54627895). This example is interesting because the phase space of the Standard Map is the surface of a torus, not a euclidean plane.

Although the orbit recognition algorithm is originally designed with a euclidean phase space in mind, it turns out the same algorithm works for the toroidal phase space. The only change that is required is to use a different distance metric in the Prim's algorithm

for constructing the minimal spanning tree. Instead of a euclidean distance metric, a torus distance metric is used to compute distances among points. Neither the orbit classification rules nor the implementation of the identification routine and the stability-control routine needs to be modified.

The snapshot in Fig. 3.18 displays the output of the orbit recognition algorithm. In panel (a) is an orbit segment of 512 iterates. We can clearly perceive a 2-loop separatrix. Note that the toroidal surface is represented by a unit square with the left and right edges, and the top and bottom edges identified. Panel (b-1) shows the MST representation of the orbit segment. The two long horizontal lines in the middle of the picture indicate that the two “ends” of the separatrix are in fact connected. In panel (b-2), we blow up a small region around the right hyperbolic point. A few non-primary nodes are seen. The bottom panel contains the symbolic description of the branching structure; it shows that the MST has a hyperbolic structure.

## **3.7 EVALUATING THE PERFORMANCE**

### **3.7.1 How reliable are the recognition results?**

For the thousands of orbits that have been tried, the recognition algorithm 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. Fig. 3.19 shows two examples to illustrate the robustness of the algorithm. In Fig. 3.19a-1, the set of 256 iterates is identified as a quasiperiodic orbit. Fig. 3.19a-2 below displays 50000 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. 3.19b-1. The recognition algorithm identifies this set as a 5-loop separatrix. Again, the overall shape of the orbit does not change much after 50000 iterates as shown in Fig. 3.19b-2.

### **3.7.2 What are the orbits that cannot be recognized?**

As I have alluded to in the preceding section, the recognition algorithm may give up and fail to assign an orbit category. Based on the thousands of orbits that have been



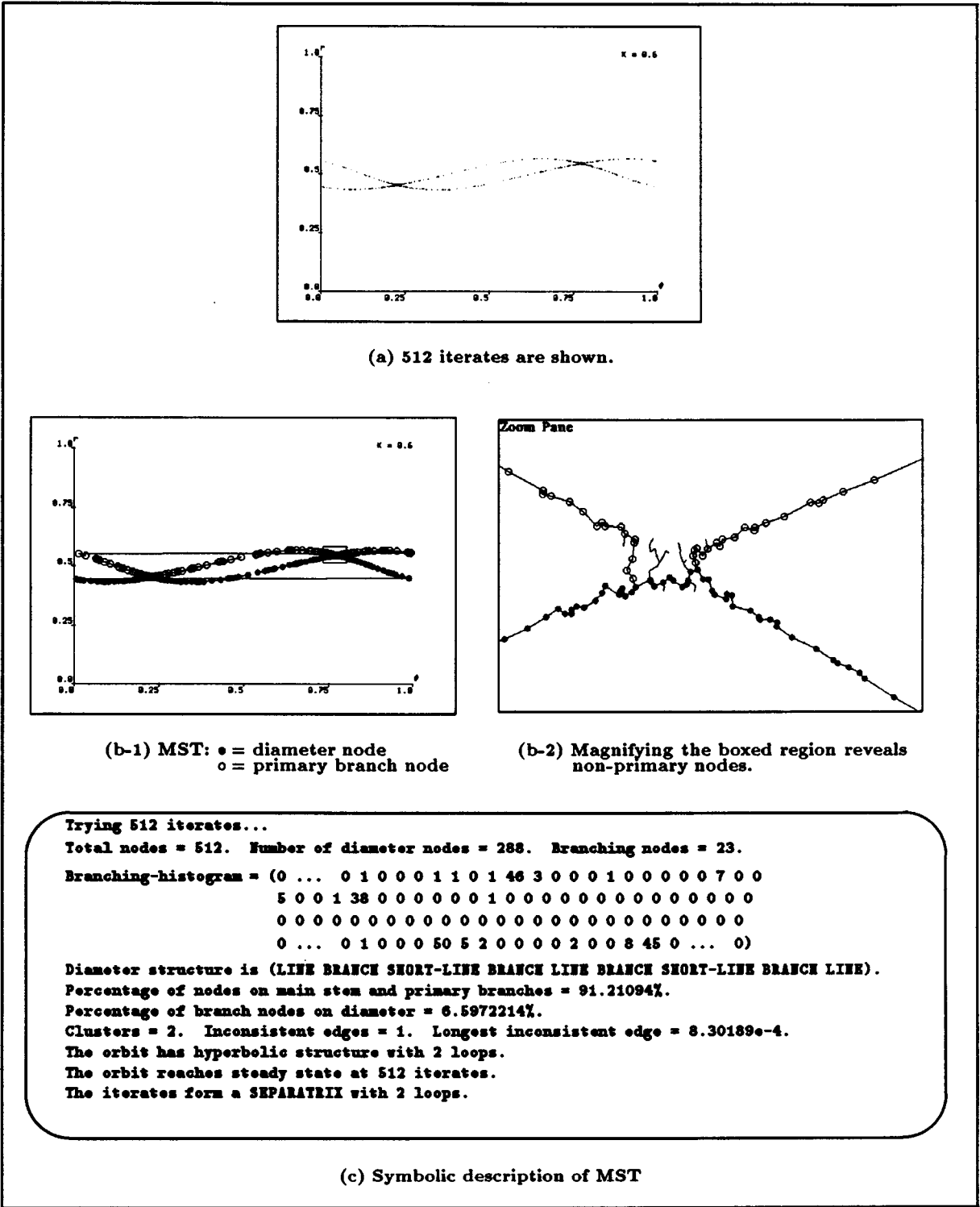


Figure 3.18: Recognizing a separatrix on the torus surface of the Standard Map. The two horizontal lines in (b-1) are result of identifying the  $\theta = 0$  and  $\theta = 1$  lines.

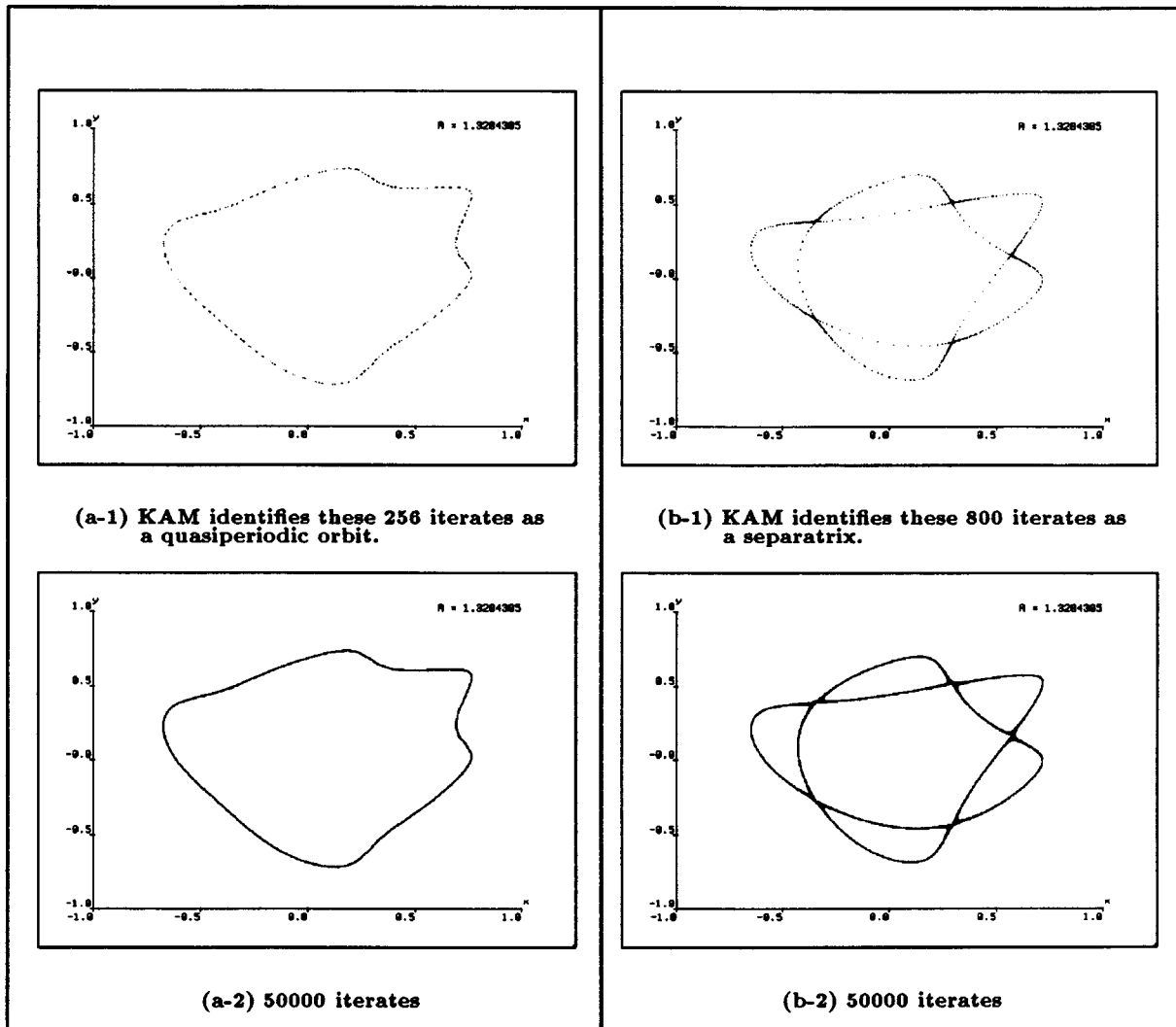


Figure 3.19: *Showing the robustness of KAM's recognition result. Both the quasiperiodic orbit (a) and the separatrix (b) retain their shape after 50000 iterates.*

experimented, the failure rate is roughly 20%, i.e., KAM fails to classify one out of every five orbits. Three types of orbits give KAM most difficulty: (1) a quasiperiodic orbit with rotation number close to a rational, (2) a secondary island chain, and (3) a separatrix. Let me discuss each of them in turn.

Informally, the rotation number  $\rho$  of a quasiperiodic orbit measures the average angle points are rotated under the iteration of a map. <sup>6</sup> If  $\rho = \frac{m}{n}$  is rational, then the orbit will repeat itself after  $n$  iterations, i.e., the orbit is periodic with period  $n$ . On the other

<sup>6</sup>Rotation number will come up again in Chapter 4.

hand, if  $\rho$  is irrational, the orbit will *asymptotically* cover a smooth closed curve densely. The catch to it is the covering is dense only in the limit as the number of iterates tends to infinity. The problematic case arises when  $\rho$  is *close*<sup>7</sup> to a rational. In such case, the initial orbit segment may spend most of its time in isolated regions in the phase space, resembling a periodic orbit. It will take a large number of iterates before the “gaps” between these isolated segments are eventually covered. An example of this is shown in Fig. 3.20a. Looking at the first 1000 iterates (Fig. 3.20a-1), we may think the orbit is periodic with period 19. But, the orbit turns out to be quasiperiodic with  $\rho$  close to  $\frac{4}{19}$  as can be seen from the 30000 iterates in Fig. 3.20a-2 and the magnified picture in Fig. 3.20a-3.

The second type of orbit that KAM fails to recognize is the secondary island chain. An example will make this clear. Consider the orbit in Fig. 3.20b-1. It has 1250 iterates, and appears to be a 21-island-chain. On closer inspection, we find that each of the “islands” actually consists of seven smaller islands (Fig. 3.20b-3). Since secondary island chains typically occupies small phase space region, they do not affect the overall dynamics significantly. Therefore, I have not implemented this particular orbit category.

The last type of troublesome orbit is a separatrix. Intuitively, it is clear why a separatrix is problematic: it is sort of intermediate between a quasiperiodic orbit and a chaotic orbit. So the initial segment of a separatrix may look like a quasiperiodic orbit for a long time. For instance, one may conclude from Fig. 3.21a that the first 1250 iterates come from a quasiperiodic orbit. The loops of the separatrix, however, reveal themselves after 4000 iterates. Note that the recognition algorithm does not classify the first thousand iterates as a quasiperiodic orbit because a few branch nodes (which are too small to be seen in the picture) already appear near the hyperbolic points.

### 3.8 CONCLUSION

This chapter shows how the difficult problem of recognizing the type of an orbit from a finite set of iterates is reduced to a simpler problem of computing shape of a dot pattern from its minimal spanning tree (MST) representation. The MST representation is a useful idea to solve the orbit recognition problem because it has three advantages: (1) it has desirable theoretical properties (such as symmetry, non-emptiness, and invariance),

---

<sup>7</sup>A standard way to measure closeness is to use the continued fraction approximation to an irrational [10].

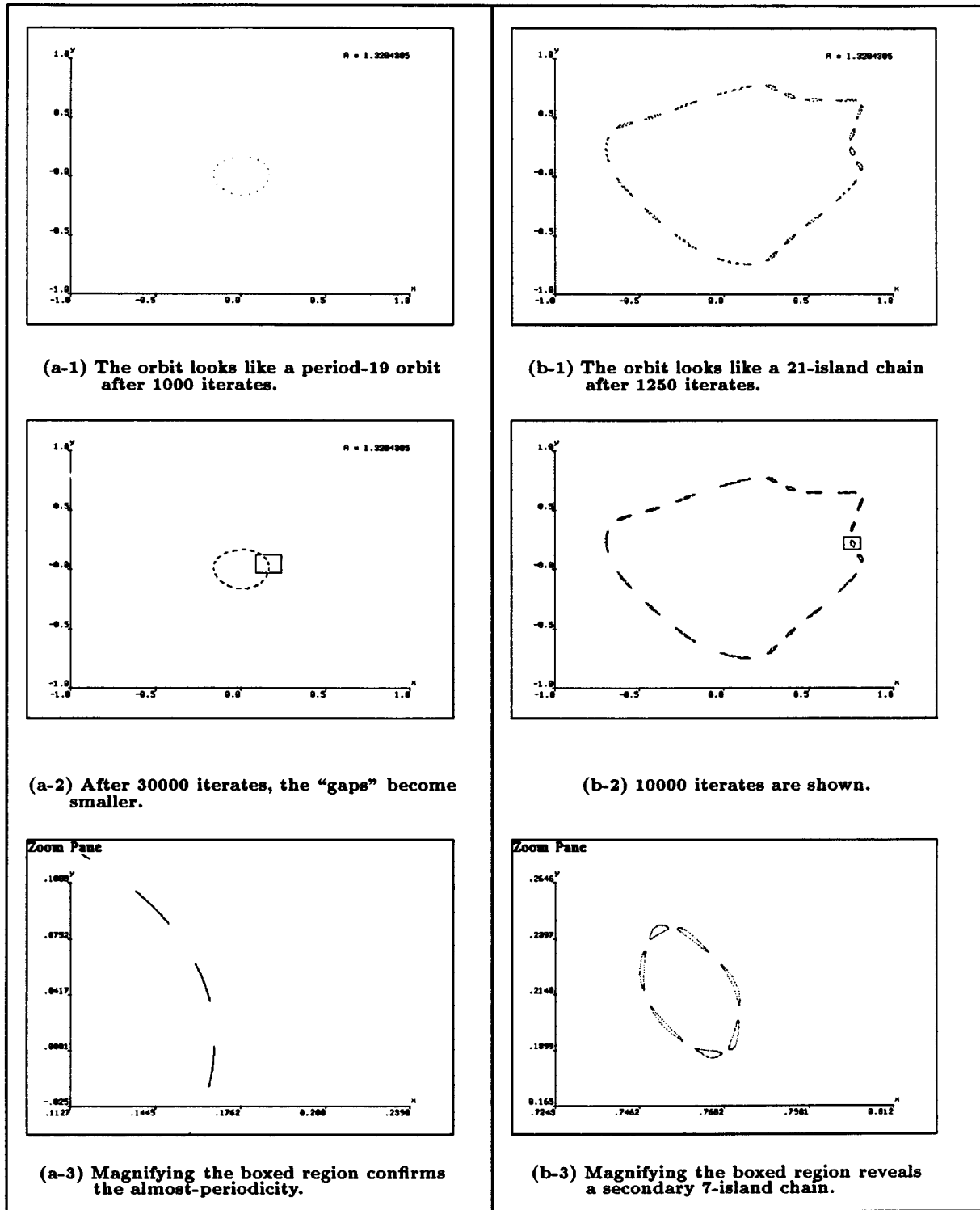


Figure 3.20: Two orbits that KAM fails to recognize: (a) on the left side is a quasiperiodic orbit with rotation number close to  $4/19$ , and (b) on the right is a secondary island chain with  $21 \times 7$  islands.

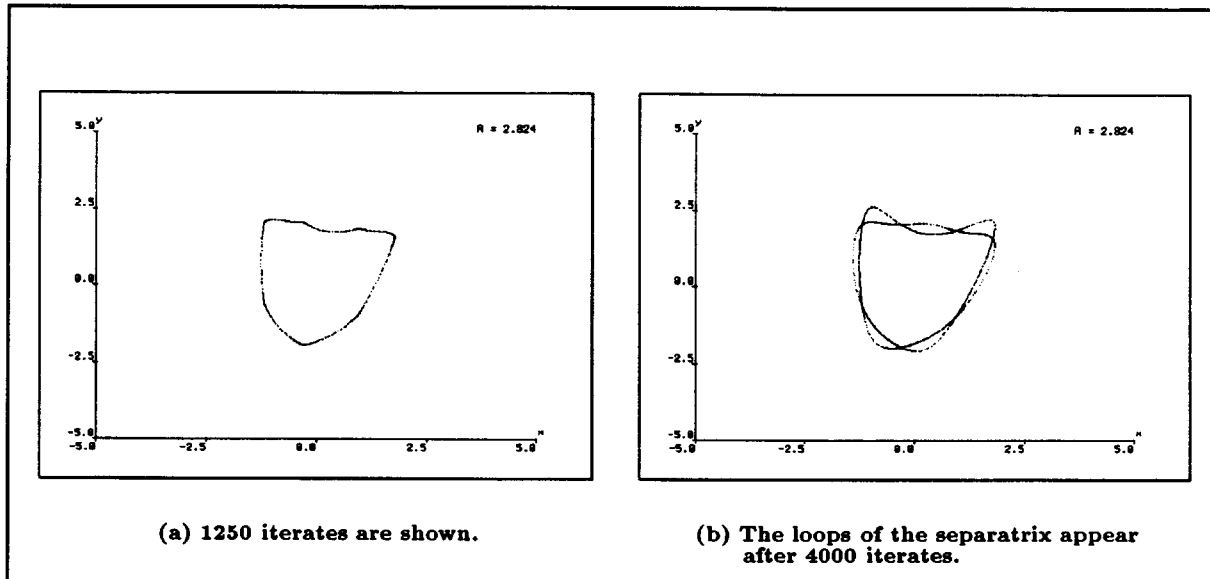


Figure 3.21: *KAM failed to recognize a separatrix with  $x_0 = (-0.31265157, -1.9546225)$  and  $A = 2.824$ . It takes over 4000 iterates to reveal the entire separatrix.*

and satisfies three constraints – soundness, connectivity, and separation – on grouping, (2) there are efficient algorithms to compute the MST of a dot pattern (e.g., Prim’s algorithm), and (3) there are domain-independent ways to compute shape of a dot pattern from its MST representation (such as Zahn’s clustering method, and the method of computing the symbolic description of the branching structure).

Knowledge relating the shape of an orbit and its orbit type is simply expressed in five orbit classification rules. That KAM is able to reliably recognize the type of an orbit is the outcome of the robustness of the symbolic description of the geometric structure of the MST. Once the orbit reaches a steady state, the clustering and branching structure of its MST representation will be stable against further addition of iterates.

Finally, we also see that the **aggregate-partition-classify** paradigm provides a simple description of the orbit recognition algorithm. In the first step, isolated iterates are aggregated into a large structure – the MST. Then, the MST is partitioned into meaningful pieces (such as an island or a loop of a separatrix). And the last step is the assignment of orbit category based on specific domain semantics. We shall see later that the aggregate-partition-classify paradigm is quite general: it is applicable to not only to the problem of orbit recognition, but also the problem of phase space searching, and the problem of parameter space searching.

## Chapter 3 Summary

Classifying the type of an orbit is important because KAM must interpret what it “sees” in order to direct the course of its numerical experiments. Orbit recognition is the problem of (1) assigning a type to an orbit, and (2) symbolically describing the qualitative features of the orbit (such as the number of islands of an island chain or the number of loops of a separatrix).

KAM's orbit recognition module implements a domain-independent clustering method based on the MST representation of a set of iterates. Isolated iterates are first **aggregated** into a MST which represents the adjacency relationships among points. Symbolic description of the branching structure of the MST is also computed. Then, the MST is **partitioned** into meaningful sub-pieces by Zahn's clustering method. The final step is to run a list of orbit classification rules which **classifies** the orbit type based on the description of the shape of the dot pattern.

Depending on the choice of the distance metric used in the Prim's algorithm, the basic algorithm is able to run both on a euclidean phase space and on a toroidal phase space. The orbit recognition algorithm has been successfully tested with thousands of orbits generated by the Henon Map, Bak Map, and Standard Map. Empirical results show that the recognition algorithm never gives a wrong answer, but can fail to give an answer 20% of the time.

# Chapter 4

## Phase Space Searching

In the previous chapter, we saw how KAM identifies the type of an orbit by exploiting the properties of minimal spanning trees. This chapter presents in detail the solution to the **Phase Space Search Problem**, which has three parts:

- How are initial states chosen?
- When is searching in phase space terminated?
- How are the search results interpreted and summarized?

Given an initial point in the phase space, a discrete dynamical system generates a sequence, possibly infinite, of points – an orbit of a specific type. Through judicious sampling of initial points, a human expert can efficiently map out the global behavior of the system. The issue is how KAM can mimic the expert's successive choices of initial points to reveal the global pattern in the phase portrait.

Simply stated, the intuitive idea behind the solution is to implement a *local consistency test* between neighboring orbits. If two adjacent orbits are compatible, there is no need to search in the region between these two orbits. On other hand, if the two orbits are incompatible, one must look further inside the region to discover missing orbits that will restore compatibility among adjacent orbits.

KAM represents a collection of orbits by an **Orbit Adjacency Graph**: each node of the graph stands for an orbit, and arcs linking two nodes denote spatial relationships (such as adjacency, inside, and outside) between the two orbits.

For instance, given some initial state, can we tell how the system is going to behave? If the pendulum is started with a slightly different initial state, how does the new behavior compare with the old one? And finally, if you pick some initial state at random, what is the likely result?

There is a classical way to answer these questions: find an explicit formula for the solutions of the differential equations. But there are problems with this approach. The first problem is that most systems do not yield to such treatment. Another problem is that even if you do find such formula, it is often very unilluminating. For instance, our simple pendulum has an explicit solution, which can be expressed in terms of some elliptic functions. But we still have to play with the formula quite a bit to unravel the information that it contains.

But there is a better way – more geometrical, more qualitative, more insightful – to find out how a pendulum behaves. This method uses the idea of **phase space**, and it comes from Poincaré .

To tell what a pendulum is doing, we need two state variables: its position and velocity. Each pair of position and velocity defines a unique state. Let's represent this state by a single point in a two-dimensional space whose axes are all the possible positions and velocities. This space is called the **phase space** for the pendulum. The origin of the phase space, which is the state with zero position and zero velocity, corresponds to the rest state in which the pendulum is hanging vertically downwards.

What does the phase space of a pendulum look like? The velocity variable can go from negative infinity to positive infinity. So the velocity axis is represented by a real line. The sign of the variable just indicates which direction – clockwise or counterclockwise – the pendulum is moving. Position is most conveniently measured in terms of the angular deviation from the vertical. So the position variable can range from  $-180^\circ$  to  $+180^\circ$ . Of course,  $-180^\circ$  is the same position as  $+180^\circ$ . That means the position axis should be represented by a circle. So it is clear that the phase space of the pendulum is made up by the product of a real line and a circle – that is, an infinitely long cylinder (Fig. 4.1).

Now imagine you start the pendulum at time zero at some initial state. Observe how the pendulum evolves in time. Measure and plot the current state of the pendulum at regular time intervals. Let's say you do the measuring and plotting frequently enough – once every half second, perhaps. You will see lots of closely-spaced points, and they trace out a curve in the phase space. This is the **orbit** starting from one particular initial state. We have seen this before in chapter 3. If the orbit looks like a circle near the origin, then you know the system is swinging back and forth.



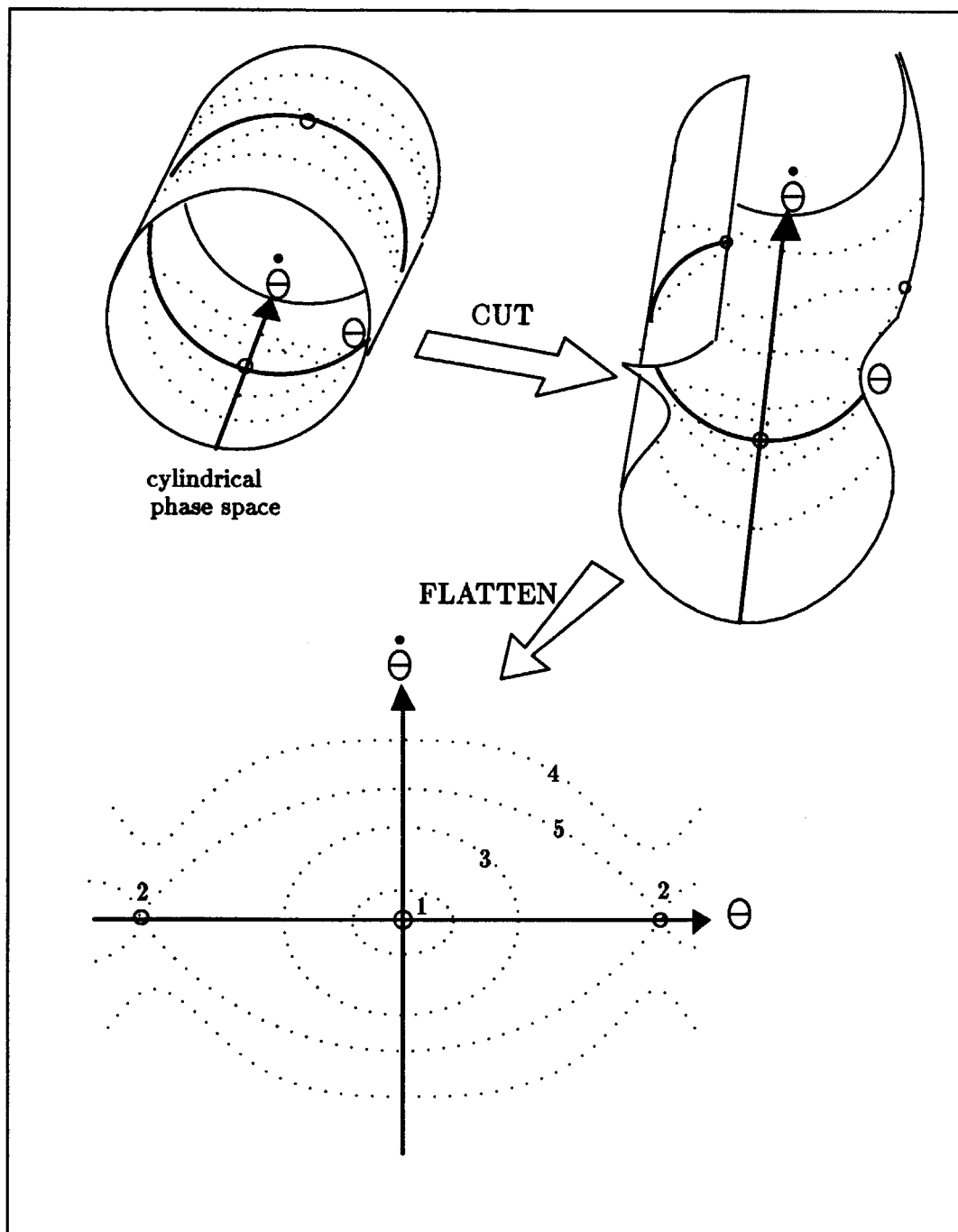


Figure 4.1: The phase space  $(\theta, \dot{\theta})$  of a simple pendulum is an infinitely long cylinder because  $\theta$  is an angular variable modulo  $2\pi$ . Performing many experiments with the pendulum will reveal the typical structure of the phase portrait. It is easier to visualize trajectories in the phase space by cutting the cylinder along the top, and flattening it as shown.

Pick a different initial state, and you get a different orbit. Sample a dozen or so initial states, and plot the resulting orbits. What will you see? If you do it right, you should get a picture like the one shown in the lower part of Fig. 4.1. The picture is rather like an eyeball that comes with eyebrows above and below. It has a few features worth exploring.

There are concentric circles (labeled 3) near the origin. They correspond to small oscillations about the rest point. The origin (labeled 1) is called a **fixed point**, and it is **stable** because small perturbation does not cause the pendulum to move far away. Another name is **elliptic fixed point**, because it is surrounded by elliptical curves. Far away from the origin, there are “open” undulating curves (labeled 4), and they correspond to the rotor motion. These curves are not really “open”. They are closed curves on the surface of the cylinder as they should be because the motion is periodic with respect to the position. Between the undulating curves and the concentric circles is an interesting orbit (labeled 5). It looks like an oval with corners at two points (labeled 2). This orbit is called a **separatrix**. It is so named because it separates two different kinds of motion: inside the separatrix the motion corresponds to small oscillation; outside, large rotation. The two points at the corner are actually the same point. It refers to the upward unstable position for the pendulum. It is called a **hyperbolic fixed point**, and is **unstable** because slight perturbation can cause completely different behaviors. Notice that the separatrix does not actually cross the hyperbolic fixed points. They are really two distinct orbits. If the system starts somewhere on the separatrix, it will go closer and closer to the unstable fixed point, but never quite get to it.

What’s the point of all this talk about phase space and the pendulum? Well, it constitutes a powerful way of capturing all possible motions of the pendulum in a *single* picture. Mathematicians give this picture a name: the **phase portrait**. A phase portrait is basically a picture that shows how different orbits fit together in phase space. It is a powerful idea because it works not just for a simple pendulum, but for *any* dynamical system. It doesn’t even matter whether the system is continuous (like our pendulum) or discrete (like the Henon Map). The same technique applies. By looking at the shape of different orbits, and how they are connected up, we can answer many interesting questions about the behavior of the system *without explicitly solving any equations*.

### **How to find the interesting orbits**

How can we generate a reasonable phase portrait? After all, there are zillions of initial states that we can pick, and in principle any one of them may lead to a behavior different

from those of nearby initial states. Phase space searching is the problem of finding a set of orbits that are representative, in the sense that they completely characterize the global, qualitative behaviors of the system. For example, the phase portrait of the simple pendulum should consist of at least those orbits that are labeled from 1 to 5 in Fig. 4.1. Missing one of these representative orbits means that an important dynamical feature will go unnoticed. On the other hand, we don't want to try too many orbits because that is too costly and almost certainly will unnecessarily complicate the phase portrait.

## 4.2 CORE IDEA: CONSISTENCY OF NEIGHBORING ORBITS

Blind exhaustive testing of every point of the phase space – the space of all possible starting conditions – to find out the typical system behavior is out of the question because the number of possible starts is overwhelming (infinite, in fact). Moreover, interesting qualitative behaviors such as a homoclinic orbit often occur in so small a region in the phase space that unguided experiments will likely miss them.

One way to meet these difficulties is to exploit knowledge of the natural constraints in the physical problem. What I am going to say next can be made much more precise, but I'd rather rely on more intuitive arguments to get the main ideas across.

Using Poincaré's geometric viewpoint, we visualize the time evolution of a system as the motion of a point in a multi-dimensional phase space. The equations of motion govern how the point should move in any instant of time. We can also imagine a collection of points moving together like fluid particles flowing along. The fluid analogy is useful because Joseph Liouville proved a theorem saying that if the system is Hamiltonian, then the fluid is incompressible [3]. That means there are no sources or sink in the fluid. If you start with a volume of "fluid particles" in phase space and let it flow, you'll find that the volume will remain the same even though its shape may change drastically. Hamiltonian systems preserve volume in phase space.

An easy corollary of the Liouville theorem is the fact that the Poincaré mapping defined on a surface near a periodic orbit is **area-preserving**, i.e., if the mapping is used to transform any region of the section, its area will remain unchanged after the transformation. This is a key difference between a Hamiltonian system and a non-Hamiltonian system. We are going to exploit the incompressibility nature of Hamiltonian flow.

Fig. 4.2a 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. global, qualitative behaviors of the system. For example, the phase portrait of the simple pendulum should consist of at least those orbits that are labeled from 1 to 5 in Fig. 4.1. Missing one of these representative orbits means that an important dynamical feature will go unnoticed. On the other hand, we don't want to try too many orbits because that is too costly and almost certainly will unnecessarily complicate the phase portrait.

## 4.2 CORE IDEA: CONSISTENCY OF NEIGHBORING ORBITS

Blind exhaustive testing of every point of the phase space – the space of all possible starting conditions – to find out the typical system behavior is out of the question because the number of possible starts is overwhelming (infinite, in fact). Moreover, interesting qualitative behaviors such as a homoclinic orbit often occur in so small a region in the phase space that unguided experiments will likely miss them.

One way to meet these difficulties is to exploit knowledge of the natural constraints in the physical problem. What I am going to say next can be made much more precise, but I'd rather rely on more intuitive arguments to get the main ideas across.

Using Poincaré's geometric viewpoint, we visualize the time evolution of a system as the motion of a point in a multi-dimensional phase space. The equations of motion govern how the point should move in any instant of time. We can also imagine a collection of points moving together like fluid particles flowing along. The fluid analogy is useful because Joseph Liouville proved a theorem saying that if the system is Hamiltonian, then the fluid is incompressible [3]. That means there are no sources or sink in the fluid. If you start with a volume of "fluid particles" in phase space and let it flow, you'll find that the volume will remain the same even though its shape may change drastically. Hamiltonian systems preserve volume in phase space.

An easy corollary of the Liouville theorem is the fact that the Poincaré mapping defined on a surface near a periodic orbit is **area-preserving**, i.e., if the mapping is used to transform any region of the section, its area will remain unchanged after the transformation. This is a key difference between a Hamiltonian system and a non-Hamiltonian system. We are going to exploit the incompressibility nature of Hamiltonian flow.

Fig. 4.2a 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 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, more likely, the velocity has some non-zero radial component. Let us suppose the velocity vector goes outward (Fig. 4.2b). 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 another half (see Fig. 4.2c). 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 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. 4.2d). 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. 4.2e). 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. 4.2f shows 6 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. 4.2). Notice that we have arrived at this non-trivial 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.

Now, let us turn this physical insight around, and use knowledge about possible local flow patterns to derive an understanding of the global behaviors of a physical system.

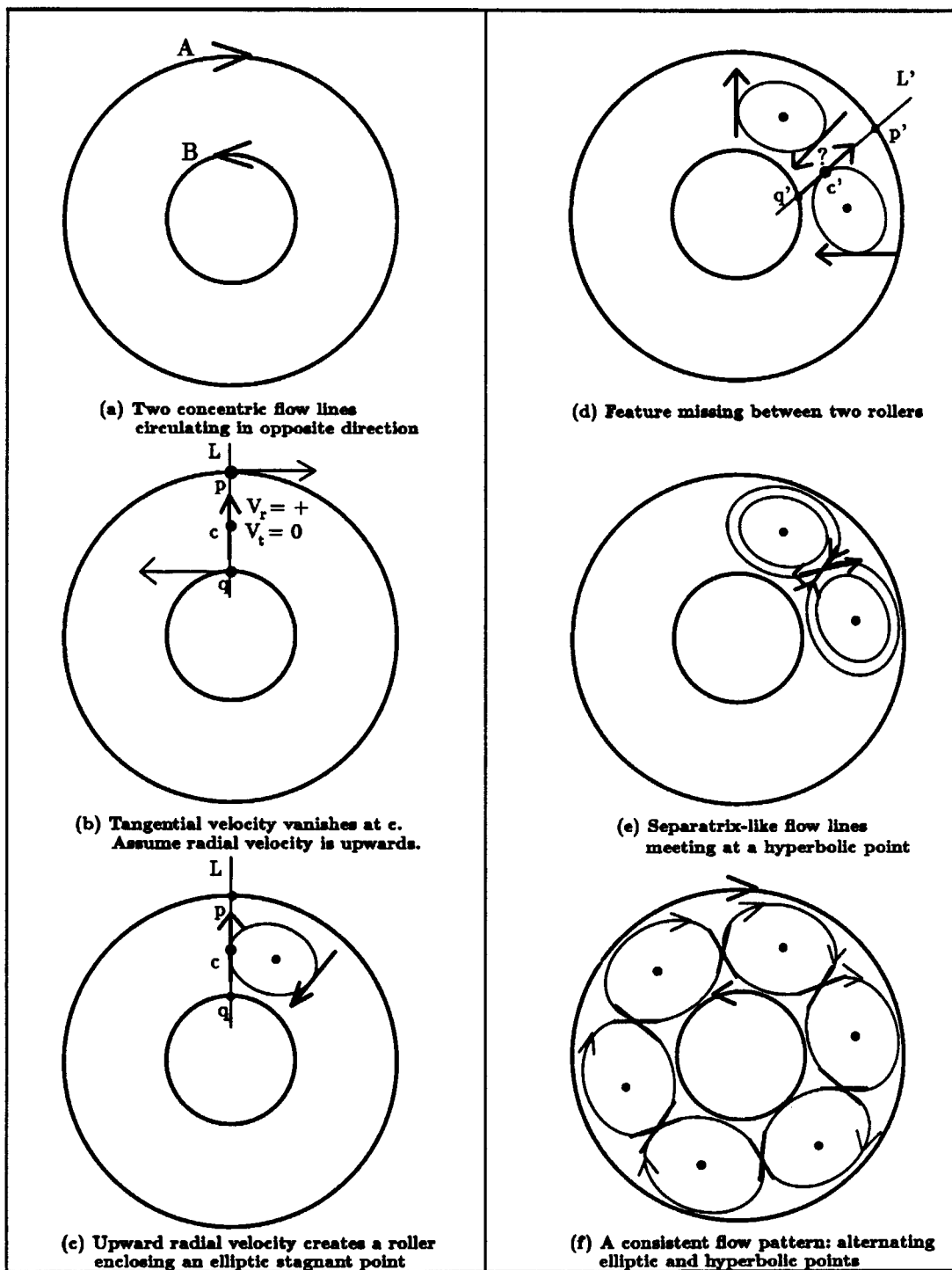


Figure 4.2: Deducing missing features from incompatibility of two neighboring incompressible flow lines.

## 4.3 ROTATION NUMBER: CONTINUITY OF DISCRETE FLOW

The consistency argument, we have just seen, depends on the fact that neighboring flow lines must satisfy a continuity requirement: a continuous flow cannot have a sudden change in flow direction. However, an orbit of a discrete mapping is not a smooth line, but consists of points jumping from one place to another. We need a new technical concept – the **rotation number** of a map – to formulate a continuity constraint for a discrete flow.

### 4.3.1 Rotation Number Defined

The best way to understand what the rotation number means is to consider a homeomorphism  $f$  (i.e., both  $f$  and its inverse  $f^{-1}$  are continuous) acting on a circle. We further suppose  $f$  is **orientation-preserving**:  $f$  preserves the cyclic order of points on the circle. So, if  $x$ ,  $y$ , and  $z$  are points on the circle with the cyclic order  $x < y < z$ , then their images under  $f$  have the same cyclic order, namely,  $f(x) < f(y) < f(z)$ . This order-preserving property severely constrains the dynamics of  $f$ , and allows an important topological invariant – the **rotation number** of a map  $f$  – to be well-defined.

Intuitively, the rotation number of  $f$  measures the average angle through which points are rotated under one application of  $f$ . It is convenient to express the angle in units of  $2\pi$  so that the rotation number takes a value between 0 and 1. Let me be more precise. Pick some point  $x$  on the circle, and measure the average angle  $\theta(n, x)$  turned through by  $x$  under  $n$  successive applications of  $f$ . See Fig. 4.3. To avoid ambiguity caused by integer multiples of  $2\pi$ , all the angles and the summation are taken modulo  $2\pi$ . Now, define the rotation number of  $f$  as follows:

$$\rho(f) \equiv \lim_{n \rightarrow \infty} \theta(n, x)$$

Note there is no reference to the point  $x$  in the notation  $\rho(f)$ . This is because the limit on the right hand side is independent of the choice of initial point. To see this, consider any point  $y$  on the circle. The sequence  $\{\theta(n, x) - \theta(n, y)\}$  will tend to zero as  $n \rightarrow \infty$  since, otherwise, one point will eventually overtake the other under the applications of  $f$ , and this is impossible given that  $f$  preserves order on the circle.

Harder to show is the fact that the limit defined above does in fact exist. Technically, it amounts to proving  $\{\theta(n, x)\}$  forms a Cauchy sequence on the real line [7]. The intuition

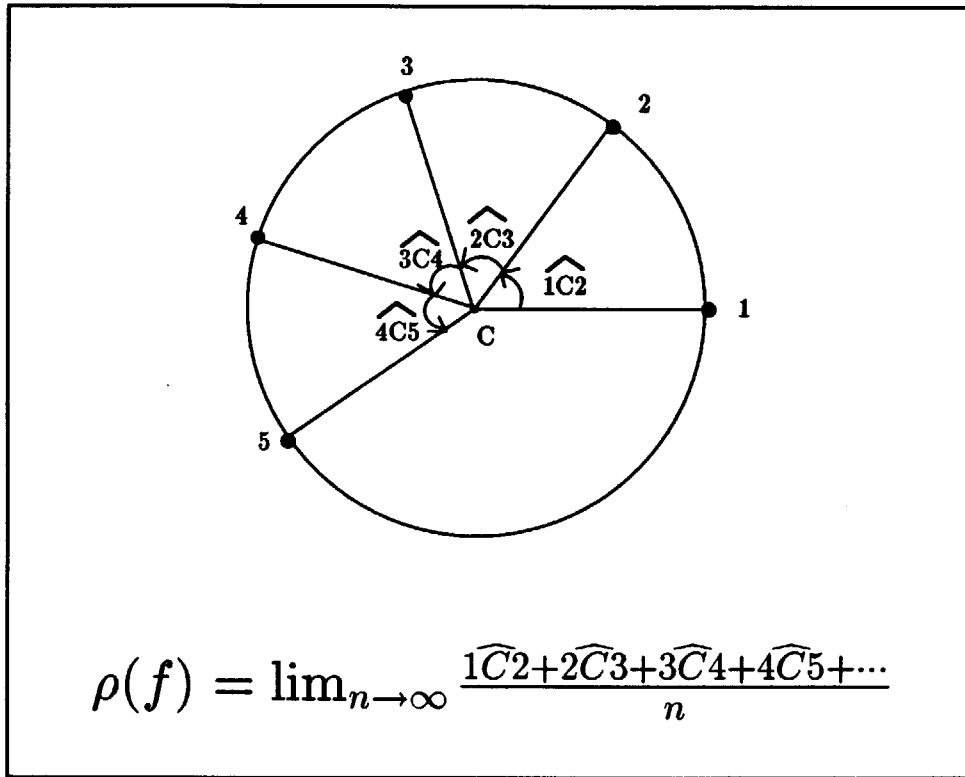


Figure 4.3: A closed curve is schematically shown as a circle around the fixed point  $C$ . Rotation number for a map  $f$  is defined as the asymptotic average angle between  $f$ 's successive iterates  $1, 2, 3, \dots$

behind the proof is similar to that used in the above argument to show the rotation number is independent of the choice of  $x$ . Briefly one argues that  $\theta(n, x)$  converges as  $n$  becomes infinite, since otherwise the point  $x$  will overtake itself on the circle. I shall not dwell on the details, but hope it is at least plausible to believe in a rotation number  $\rho(f)$  such that asymptotically the average angle turned by any point on the circle is  $2\pi\rho(f)$ .

A useful property about the rotation number is that it is *invariant* under coordinate transformation – any continuous, invertible transformation – of the circle. This property allows us to talk about the rotation number of an orientation-preserving homeomorphism acting on any curve topologically conjugate to a circle.



### 4.3.2 Compatibility of Rotation Numbers

Why is the rotation number a useful concept? The main reason is that it gives a nice characterization of a quasiperiodic orbit. As we recall from Chapter 3, topologically a quasiperiodic orbit covers a circle densely. It is clear that its rotation number can't be rational because that will give rise to a periodic orbit. For if the rotation number is, say,  $\frac{m}{n}$ , then any initial point will come back to its exact starting position after  $n$  applications of the mapping. That means the orbit is periodic. So the rotation number of a quasiperiodic orbit must be irrational [7]. The converse – that irrational rotation number must give rise to a quasiperiodic orbit – is also true provided that the mapping involved is sufficiently smooth.<sup>1</sup> So for each quasiperiodic or periodic orbit, we can assign a unique rotation number to it. The rotation number of a quasiperiodic orbit is always irrational; for a periodic orbit, it is rational.<sup>2</sup>

Another useful property of the rotation number is that it changes continuously with respect to initial states, i.e., a small change in the initial state causes a small change in the rotation number.<sup>3</sup> The continuity of rotation number gives us a very important constraint. In particular, it tells us where to locate periodic orbits and island chains. Consider two nearby quasiperiodic orbits having rotation numbers  $\rho_1$  and  $\rho_2$  respectively. Let's suppose  $\rho_1$  is slightly smaller than  $\frac{1}{5}$ , and  $\rho_2$  slightly larger. By continuity, there must exist a third, nearby orbit with rotation number exactly equal to  $\frac{1}{5}$ . In other words, a periodic orbit of period 5 must exist between the two quasiperiodic orbits (Fig. 4.4). This also implies the existence of an island chain surrounding the period-5 orbit (section 4.2).

But there is one problem. There are infinitely many rationals between any two irrationals, and so infinitely many periodic orbits must exist between the two quasiperiodic orbits. This is in fact true. However, most of these periodic orbits have tiny island chains surrounding them. That the size of an island chain is inversely proportional to the number of the islands should be clear because after all the more islands you have the less space each island has. The number of islands grows linearly, but their area shrinks quadratically [3, Appendix 7]. So island chains with large periods will be tiny,

---

<sup>1</sup>The class of  $C^2$  mappings is sufficient. A  $C^2$  mapping is a continuous mapping whose first and second derivatives exist and are continuous.

<sup>2</sup>To many, the distinction between rational and irrational number seems nonphysical. Yet, in the context of dynamical systems, the rationality of a rotation number makes a qualitative difference in the nature of the orbit in question.

<sup>3</sup>This statement has been proven for any area-preserving map that is sufficiently close to an integrable area-preserving map [16]

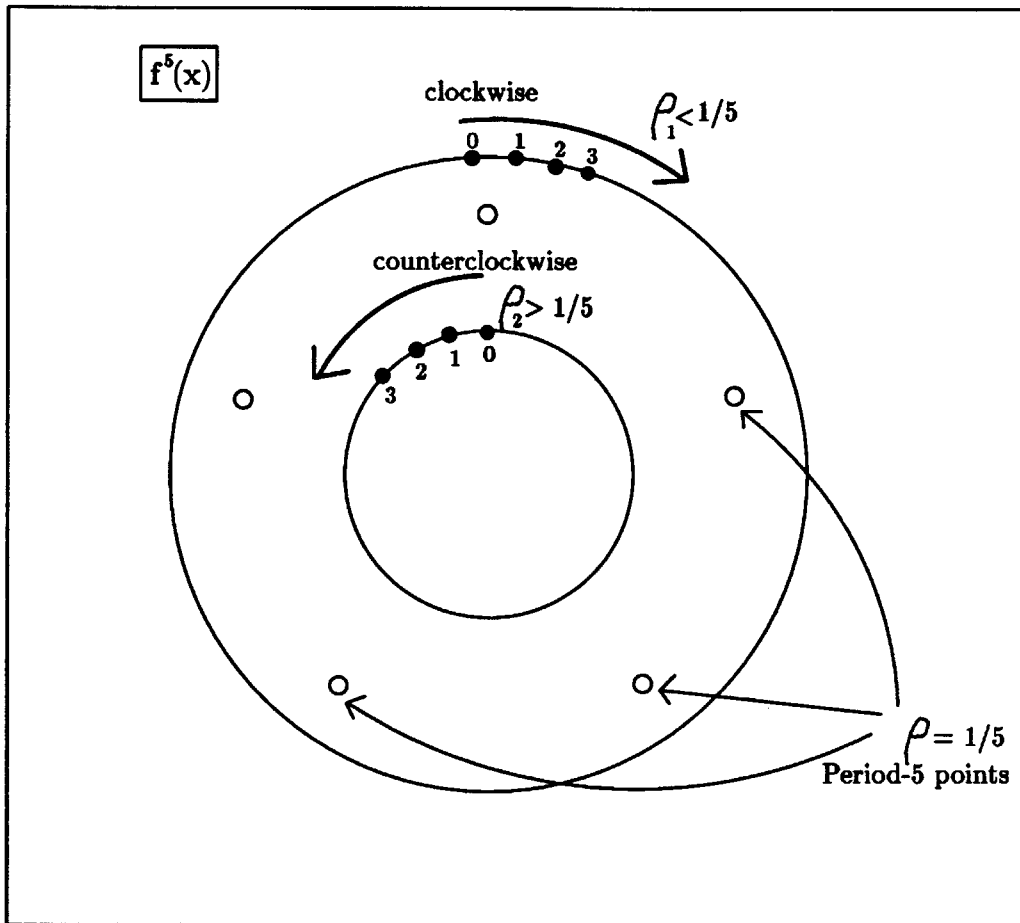


Figure 4.4: Looking at every  $n$ th iterate of  $f$  generates a sense of direction for the discrete flow. Two quasiperiodic orbits in the neighborhood of a period-5 orbit are shown. The fifth iterates of the outer orbit rotate clockwise because its rotation number  $\rho_1$  is less than  $\frac{1}{5}$ . On the other hand, the fifth iterates of the inner orbit rotate counterclockwise because its rotation number  $\rho_2$  is greater than  $\frac{1}{5}$ .

and therefore for most purposes they do not significantly affect the overall dynamics of the system.

KAM does not compute the rotation number of a quasiperiodic orbit exactly, because for phase space searching it is more informative to bound the rotation number of the orbit by some rationals whose denominators are small integers. For example, if a quasiperiodic orbit has a rotation number bounded by, say,  $(\frac{1}{5}, \frac{1}{4})$ , then it is reasonable to look for a period-4 or period-5 orbit in the neighborhood of this orbit. In other words, the bound

of a rotation number tells us not only where to look for a missing orbit, but also what type of orbit to be expected. Consequently, the rotation number of a quasiperiodic orbit is represented by a bound, an open interval with rational endpoints. I use the notation  $\rho \in (p \ q)$  to mean that the rotation number  $\rho$  lies somewhere between the irreducible rationals  $p$  and  $q$ .<sup>4</sup>

One final problem remains: How do we choose a bound for the rotation number of a quasiperiodic orbit? This question requires some mathematical techniques from number theory because in some sense we are really asking the question “What is a good rational approximation to an irrational?”.

Dynamicists have discovered that a gadget from number theory – the Farey sequence – explains nicely the order in which island chains appear. A Farey sequence of order  $n$  consists of all rational numbers of the form  $\frac{p}{q}$ , where  $q \leq n$ , in the interval  $[0,1]$  arranged according to their magnitude [10]. The definition is only defined for positive integer  $n$ . For example, a Farey sequence of order 5 is the following sequence of rational numbers:

$$\frac{0}{1}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{1}{1}$$

It is very easy to compute a Farey sequence. For instance, the following simple recursion will generate a Farey sequence of order  $n$ :

$$p_{k+2} = \left\lfloor \frac{q_k + n}{q_{k+1}} \right\rfloor p_{k+1} - p_k$$

$$q_{k+2} = \left\lfloor \frac{q_k + n}{q_{k+1}} \right\rfloor q_{k+1} - q_k$$

where  $p_0 = 0, q_0 = p_1 = 1, q_1 = n$  and  $\frac{p_k}{q_k}$  is the  $k$ th element of the sequence.

How is the Farey sequence used? Well, given that we are interested in large island chains (i.e., those with small periods), we will choose the bounds for rotation numbers from a Farey sequence of order  $n$  for some small  $n$ . In my program, I have set  $n$  to 30. That means as far as phase space searching is concerned, I will ignore all those island chains that have more than 30 islands. The Farey sequence motivates the following definitions of compatibility of rotation numbers:

**Definition 3** *Two rotation numbers are compatible if there is no rational from a Farey sequence of order 30 that lies between their associated bounds.*

---

<sup>4</sup>An irreducible rational is a rational whose numerator and denominator have no prime factor in common.

So, for example,  $\rho_1 \in (\frac{1}{6} \frac{1}{5})$  is incompatible with  $\rho_2 \in (\frac{1}{5} \frac{1}{4})$  because  $\frac{1}{5}$  lies between the two bounds.

Later we also want to talk about whether a periodic orbit is consistent with a neighboring quasiperiodic orbit. That means we need to say when a period, which is an integer, is compatible with a rotation number. Here is the definition:

**Definition 4** *An period  $k$  is compatible with a rotation number  $\rho$  if  $k$  is equal to the denominator of one of the endpoints of the bound for  $\rho$ .*

For example, an island chain with period 5 will be compatible with a quasiperiodic orbit whose rotation number is bounded by  $(\frac{1}{5} \frac{1}{4})$ .

## 4.4 ORBIT COMPATIBILITY

### 4.4.1 Pairwise Orbit Consistency

The key strategy underlying KAM's search for orbits in phase space is to focus the search in regions where neighboring orbits are inconsistent. This section describes KAM's rules for determining pairwise orbit consistency. These rules capture the constraints among neighboring orbits that are derived from the physical continuity of incompressible flow, and the continuity of rotation number (section 3.2 and 3.3).

As explained in Chapter 3, KAM recognizes six primitive orbit types. For considering consistency, we can make two simplifications. First, a periodic orbit is thought of as a degenerate quasiperiodic orbit with rational rotation number. Second, neither a chaotic orbit nor an escape orbit constrains its neighboring orbits. With three orbit types – quasiperiodic, island chain, and separatrix – the number of possible pairwise combinations is six. The inconsistent combinations are described by the rules below (see Fig. 4.5):

- **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.

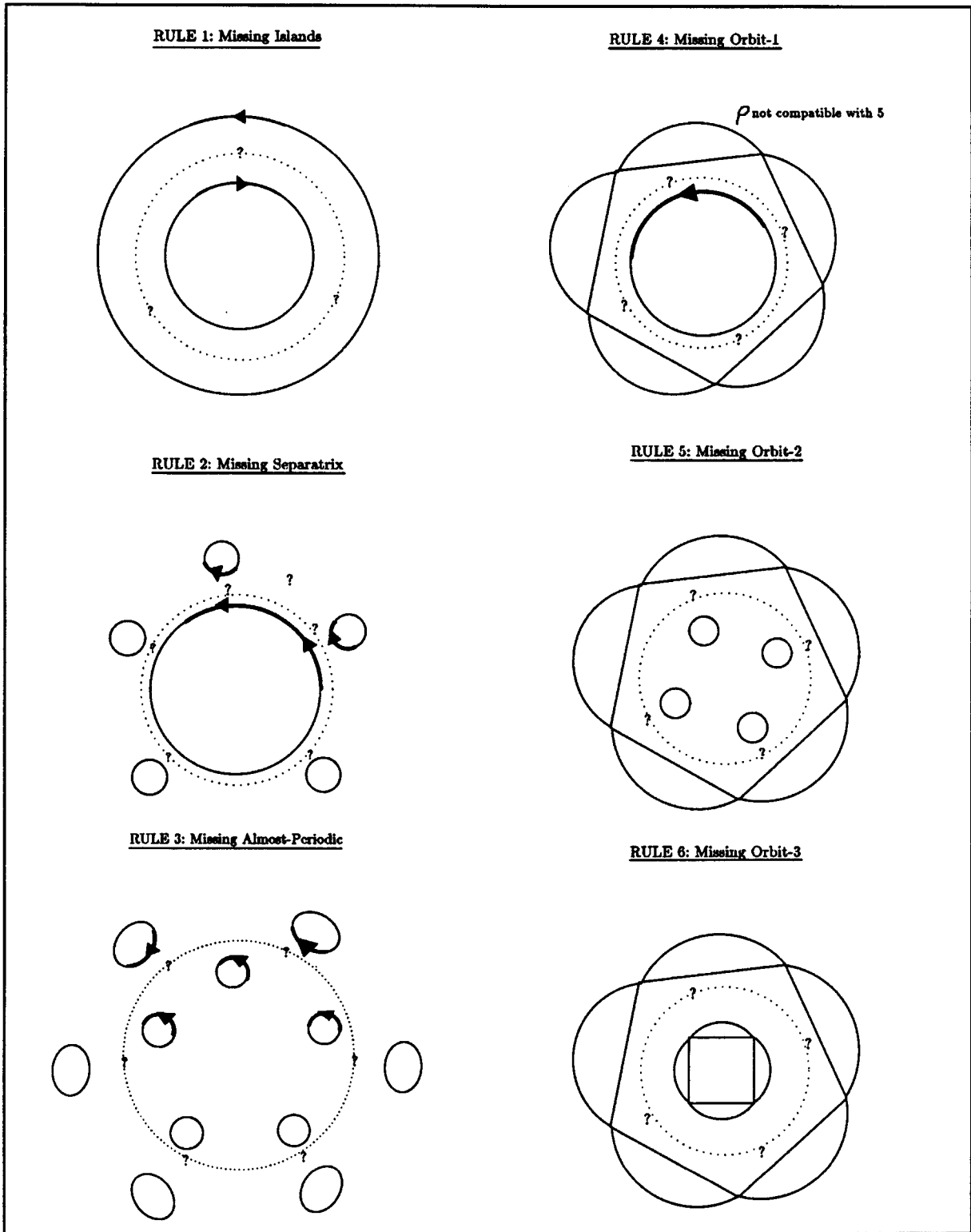


Figure 4.5: The six pairwise orbit inconsistent rules.

- **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  $\rho$  unless  $k$  is compatible with  $\rho$ .
- **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.

#### 4.4.2 Single Orbit Consistency

Besides the pairwise orbit consistency rules, KAM has rules that act on single orbits. Unlike the pairwise consistency rules, the single orbit rules are mainly heuristics. The purpose of these single orbit rules is to ensure that no large portion of the phase space remains unexplored. There are five single orbit rules:

1. **Boundary Circle Rule:** A boundary circle – a quasiperiodic orbit isolated on its outside from other quasiperiodic orbits – is inconsistent.
2. **Empty Circle Rule:** A quasiperiodic orbit that encloses a large region, but does not enclose any other orbit is not consistent.
3. **Boundary Island Rule:** A boundary island – an island chain isolated on its outside from other island chains – is inconsistent.
4. **Empty Separatrix Rule:** A separatrix that does not enclose any island chain is inconsistent.
5. **Chaotic Hole Rule:** A chaotic region – a region covered by the iterates of a chaotic or an escape orbit – containing a large empty “hole” is not consistent.

#### 4.4.3 Suggestion Rules

After an orbit incompatibility is discovered, KAM searches for the missing orbits to restore consistency. The purpose of the suggestion rules is to propose new candidate

initial states for further exploration. Each suggestion rule handles a specific inconsistency situation. Currently, KAM has a total of 12 suggestion rules – 7 of them come from pairwise inconsistency, and 5 from single orbit inconsistency.

The basic idea behind the suggestion rules is to do a “bisection search” in the region delimited by the offending orbit(s). To illustrate how the new initial states are proposed, I discuss three suggestion rules that are used most frequently. Other rules are similar in nature. The rules are named by the following convention. A suggestion rule for pairwise inconsistency is named “*type1-and-type2* suggestion rule”, where *type1* and *type2* are the types of the offending orbits. A suggestion rule for single orbit inconsistency is named after the corresponding inconsistency rule. For example, the boundary circle suggestion rule corresponds to the boundary circle inconsistency rule.

### Quasiperiodic-and-quasiperiodic suggestion rule

If two quasiperiodic orbits are inconsistent, the midpoints between the two closed curves are promising initial states to try. We can define midpoints between two closed curves C and D as follows. Let *c* be any point on C, and the closest point to *c* belonging to D be *d*. Then, we say *p* is a midpoint between C and D if *p* is the midpoint of some line segment containing such pair *c* and *d*. In the special case of two concentric circles with radii  $r_1$  and  $r_2$ , the set of midpoints is another concentric circle with radius  $\frac{1}{2} \times (r_1 + r_2)$  (Fig. 4.6).

There are many such midpoints to consider. In an early version of KAM, the suggestion rule simply randomly chooses one of these midpoints. However, later experiments show that a more effective choice is what I call the maximal midpoints which are midpoints between the orbit points where the two closed curves are (locally) maximally apart (see Fig. 4.7).

To compute locations that are locally maximally apart, I use the following procedure (see Fig. 4.7a):

1. Parameterize the first orbit by arc length.
2. Compute the distances between the two orbits as a function of arc length.
3. Smooth the distance function obtained in Step 2 by the Gaussian mask at scale 4.
4. Find the local extrema of the distance function by locating zero-crossings.

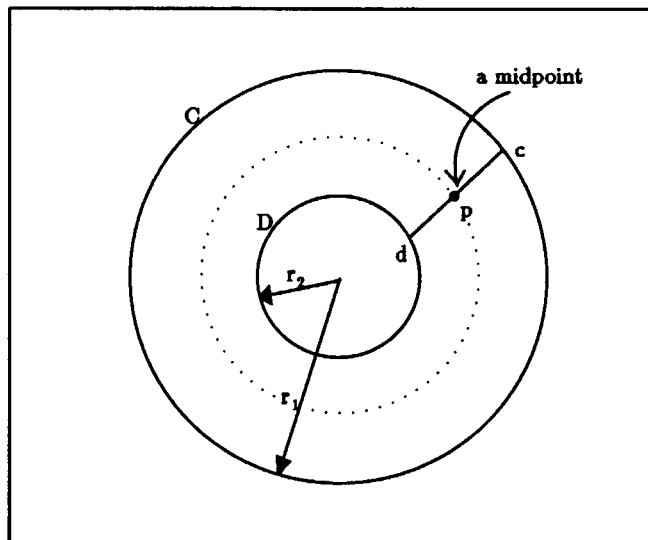


Figure 4.6: *The dotted circle represents possible midpoints between two nested closed curves.*

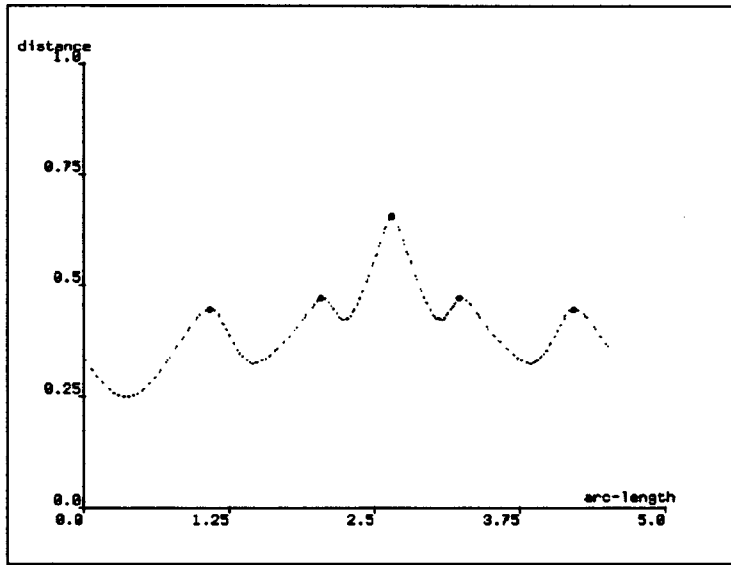
If there is an island chain, not yet seen, between the two quasiperiodic orbits, a maximal midpoint is a good approximation to the position of one of these island centers. This is because the islands are usually “oval-like”, and they are more likely to occur in a wider region between the two neighboring quasiperiodic orbits.

### Quasiperiodic-and-Island suggestion rule

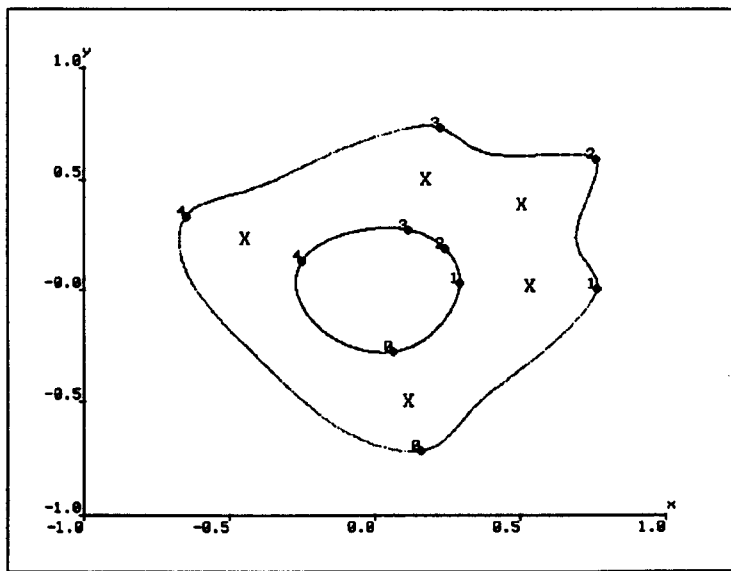
When a quasiperiodic orbit and an island chain are incompatible, typically <sup>5</sup> two candidate initial states are proposed (see Fig. 4.8). The first initial state is chosen randomly from the set of midpoints between the islands and the quasiperiodic orbit. The second initial state is chosen randomly from the set of midpoints between any two successive islands of the island chain. Midpoints between two islands are good approximations to the locations of the hyperbolic periodic points. That hyperbolic periodic points are usually located between two adjacent islands is a consequence of a common type of bifurcation of periodic orbits, and we will come to that in chapter 5.

<sup>5</sup>A special case due to the Rimmer Bifurcation is discussed in section 4.7.2.





(a) Distance between two orbits as a function of arc length.  
 (• indicates local maximum)



(b) Five pairs of points (indicated by •) that are (locally) maximally apart.

Figure 4.7: Quasiperiodic-and-quasiperiodic suggestion rule. One of the maximal mid-points (indicated by  $\times$ ) is randomly chosen.

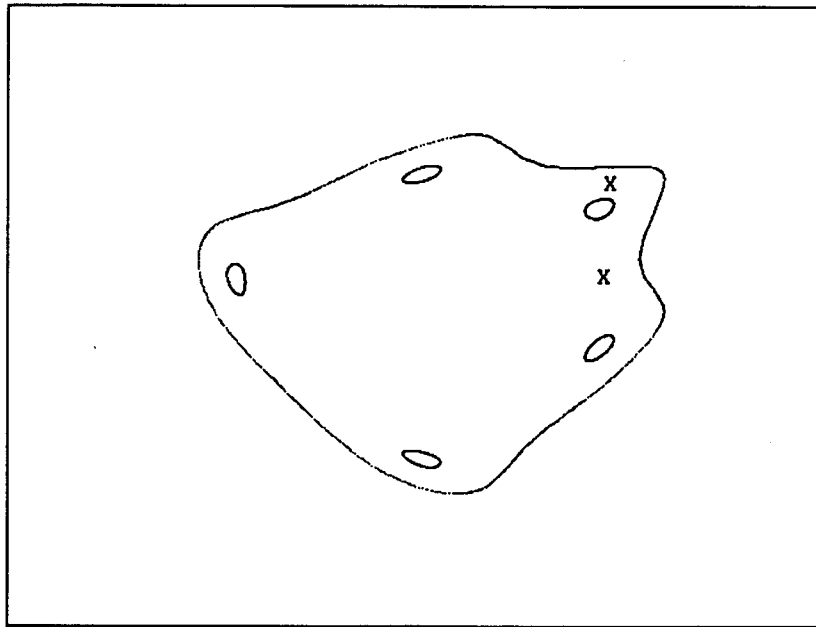


Figure 4.8: *Quasiperiodic-and-Island suggestion rule. Two candidate initial states (indicated by  $\times$ ) are proposed.*

### **Boundary-Circle suggestion rule**

Recall that a boundary circle is a quasiperiodic orbit isolated on its outside from other quasiperiodic orbits. The purpose of this suggestion rule is to extend the orbit as far out as possible. There are two cases to consider. First, the simple case, when there is no orbit outside the boundary circle, a midpoint between the orbit and the boundary of the phase space is proposed. Second, when there are orbits outside the boundary circle, the maximal midpoint is chosen by a procedure similar to that for the Quasiperiodic-and-Quasiperiodic suggestion rule.

## 4.5 IMPLEMENTATION

### 4.5.1 Representation of Phase Portrait

KAM represents the phase space by a rectangular grid. Currently, a grid of  $25 \times 25$  is used. The internal representation of a phase portrait contains information about the range of the state variables, parameter value, the list of orbits already found, and statistics related to the current simulation run. It also keeps track of the status of the grid: whether a given grid cell is empty, or occupied by an orbit.

### 4.5.2 Representation of Orbits

Each orbit is described by its type, the set of iterates, and the grid cells it occupied. Two orbits are equivalent if (1) their types are identical, and (2) they occupy the same grid cells in the same phase portrait. When KAM finds a new orbit that is equivalent to some old orbit, the new orbit is considered not useful, and will be discarded.

### 4.5.3 Computing Rotation Number Bounds

KAM computes the upper and lower rational bound of a rotation number  $\rho$  for a quasiperiodic orbit as follows. The default bound is  $(0 \ 1)$ . KAM iteratively tightens the bound by looking at successive iterates of the orbit. Specifically, the first iterate of the orbit is chosen as the initial point  $x_0$ . Let  $n$  be the number of iterates it takes for the orbit to come around once and overtake  $x_0$ . The new approximation for  $\rho$  is then  $(\frac{1}{n} \ \frac{1}{n-1})$ . The approximation is improved by keeping track of the number of iterates going around and overtaking  $x_0$  twice, three times, and so on. The approximation process terminates when one of the bounds is no longer a low-order rational.

Why does this method of computing the rotation number bound give us a correct answer? Since the rotation number is defined by an asymptotic average, isn't it possible for a map  $f$  to rotate, say, much faster than the average in the first few iterates, and then slow down in the next few to make up the same average?

Fortunately, this situation is not possible with a orientation-preserving homeomorphism  $f$ . To see this, look at Fig. 4.9. Shown in the figure is a map whose rotation number

appears to lie between  $\frac{1}{4}$  and  $\frac{1}{5}$  after the orbit comes around once. Question: Where can the image of point 5 go? Because point 5 lies on the circular arc between point 0 and 1, and because  $f$  is continuous and orientation-preserving,  $f(5)$  has to lie on the circular arc between  $f(0)$  and  $f(1)$ . We can repeat the same argument for the image of point  $f(5)$ , the image of  $f(f(5))$ , and so on. So, when the orbit comes around point 0 the second time, the point  $f^4(5)$  is going to lie somewhere on the circular arc between point 4 and 5. The point  $f^4(5)$  will be closer to point 0 than one of these two points. As a result, the rotation number bound calculated from the first two revolutions is tighter than that obtained from the first revolution alone. In fact, the rotation number bounds computed from each successive revolution form a monotonic sequence asymptotically converging to the rotation number.

#### 4.5.4 Consistency Complaints

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 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.

#### 4.5.5 The Basic Algorithm: Consistency Maintenance

The basic data structure is an Orbit Adjacency Graph. 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.<sup>6</sup> 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)

---

<sup>6</sup>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.

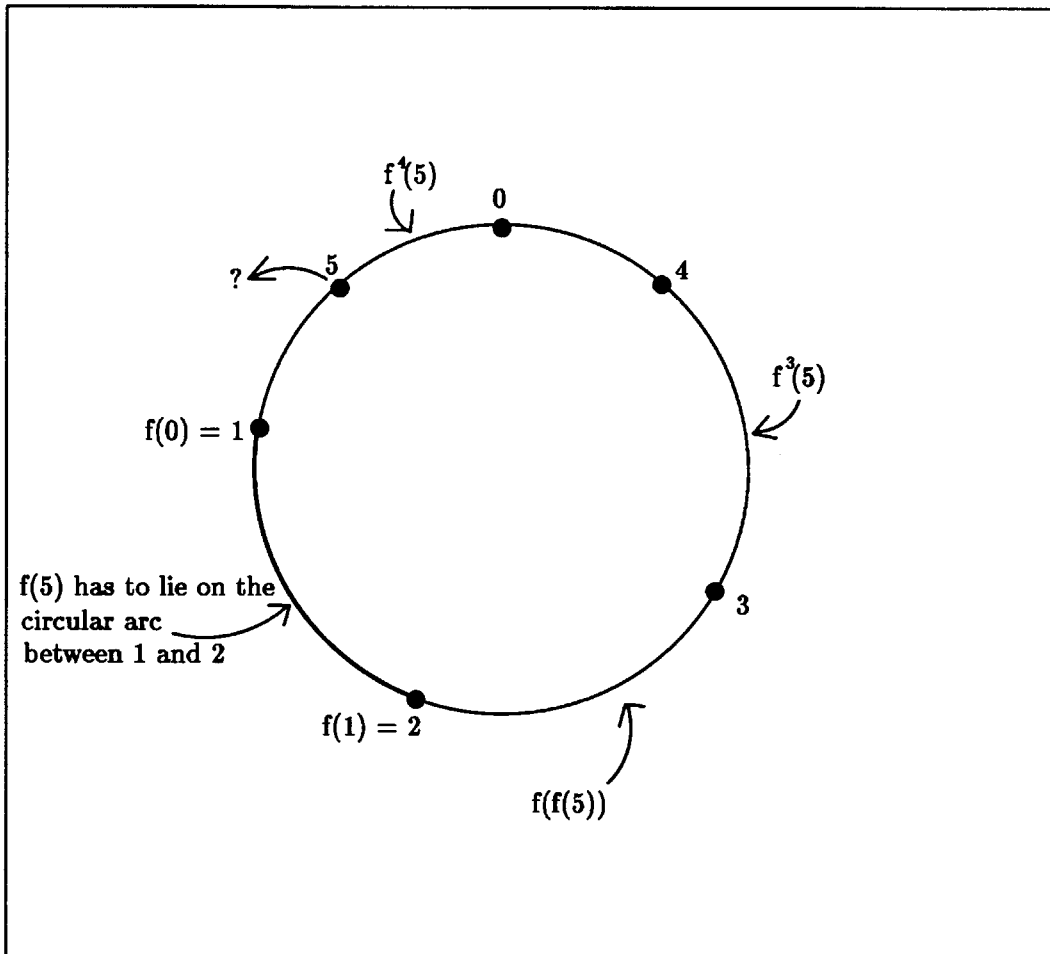


Figure 4.9: *Initial iterates of an order-preserving homeomorphism constrain the locations of the subsequent iterates. As a result, the rotation number bounds calculated from each successive complete revolution form a monotonic sequence converging to the rotation number.*

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 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:

1. Initially, pick some random initial states. Create orbits corresponding to these states.
2. Add the newly created orbits to the Orbit Adjacency Graph.
3. Update adjacency links. Produce a list of invalid adjacency links to be removed, and a list of new adjacency links to be added.
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.
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.
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.
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 <sup>7</sup>, the goal is abandoned, and the associated complaint removed.
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.

---

<sup>7</sup>An orbit is useless if either it is equivalent to some orbit previously found, or KAM fails to recognize its type.

### 4.5.6 Determining Adjacency

Roughly speaking, two orbits are adjacent if there is no orbit between them. KAM implements this idea of adjacency by a contour activation procedure. To find the neighboring orbits of a given orbit  $A$ , the grid cells surrounding orbit  $A$  in the internal representation of the phase portrait are activated. The activation spreads both outwards and inwards, iteratively activating neighboring grid cells (see Fig. 4.10). 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.

## 4.6 EXPERIMENT: MAIN ILLUSTRATION

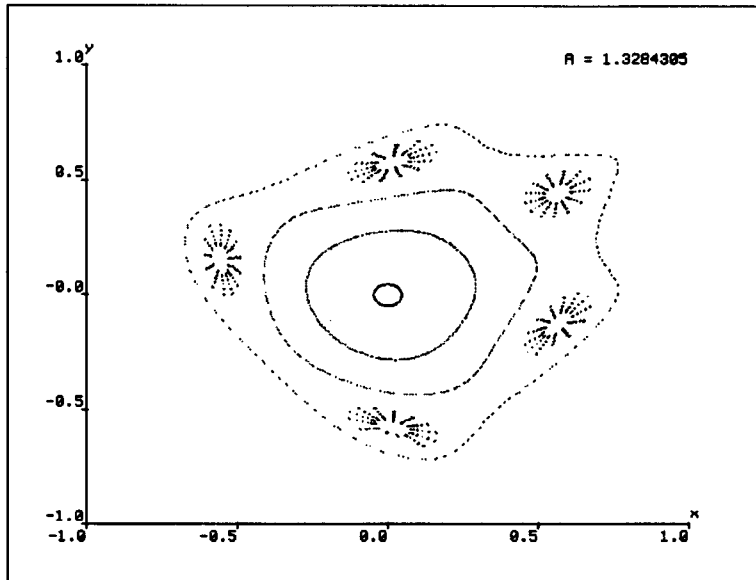
To illustrate the algorithm, I give an example of how KAM explores the Henon map with a specific parameter value  $\cos \alpha = 0.24$ . The total running time for this experiment is 3.5 hours.<sup>8</sup> KAM explores 15 initial states. Ten of these orbits are useful: 5 quasiperiodic orbits, 3 island-chains, 1 separatrix, and 1 escape orbit. Fig. 4.11 displays these orbits; the figure is remarkably similar to Fig. 4.12 taken from Henon’s paper. The remaining five initial states result in orbits that KAM fails to recognize.

Fig. 4.13 shows how KAM finds the first eight orbits. The following is a summary of the steps leading to the discovery of these orbits. Parts of the symbolic structure representing the Orbit Adjacency Graph are shown alongside the first six snapshots (Fig. 4.13a-f). Note that KAM successfully finds the separatrix at the eighth trial (see Fig. 4.13i and step 7 below).

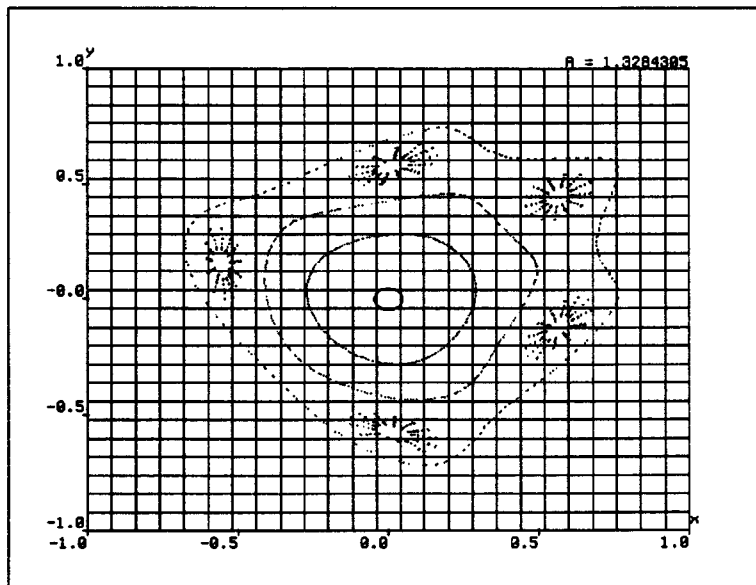
1. Pick a random initial state (0.08 0.72). Result: orbit-1 (QUASIPERIODIC) with  $\rho \in (\frac{5}{26} \frac{1}{5})$ .

---

<sup>8</sup>It takes KAM about an order of magnitude more time to produce a phase portrait than it takes a human expert. The main reason is that the human expert is much faster in pattern recognition. KAM spends roughly 80% of its time in orbit recognition.



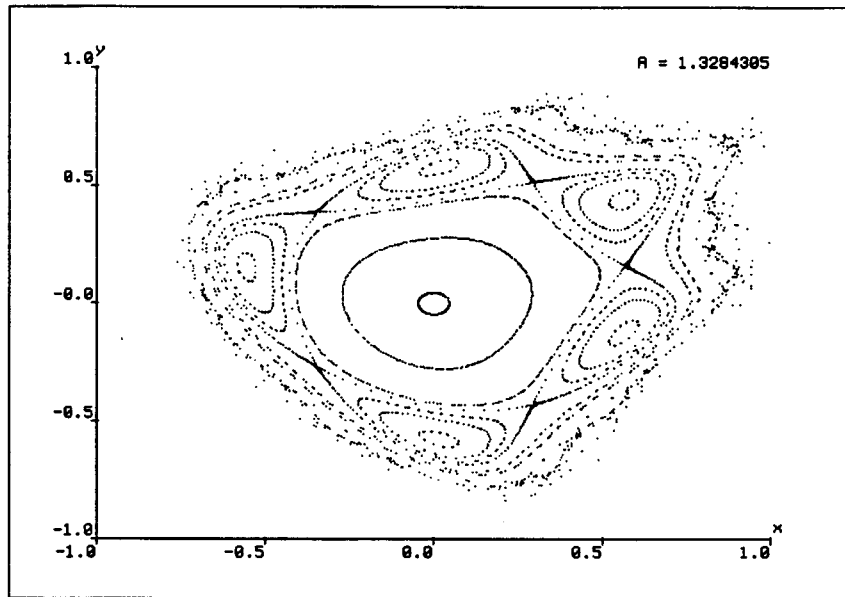
(a) Propagating the contour of the 5-island chain outwards



(b) Showing the underlying 25 x 25 grid.

Figure 4.10: An example of determining adjacent orbits by propagating contour of an orbit.





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.

Figure 4.11: Upper: KAM finds 10 orbits in the finished phase portrait; the portrait resembles that shown in Fig. 4.12. Lower: KAM generates a summary report of its findings.

2. **Empty Circle Rule.** Pick the centroid of orbit-1 at (0.04 0.01). Result: orbit-2 (QUASIPERIODIC) with  $\rho \in (\frac{4}{19} \frac{3}{14})$ .
3. **Missing Islands Rule.** Pick a midpoint (-0.23 -0.11) between orbit-1 and orbit-2. Result: orbit-3 (QUASIPERIODIC) with  $\rho \in (\frac{5}{24} \frac{4}{19})$ .
4. **Missing Islands Rule.** Pick a midpoint (0.48 0.16) between orbit-1 and orbit-3. Result: orbit-4 (QUASIPERIODIC) with  $\rho \in (\frac{1}{5} \frac{1}{4})$ .

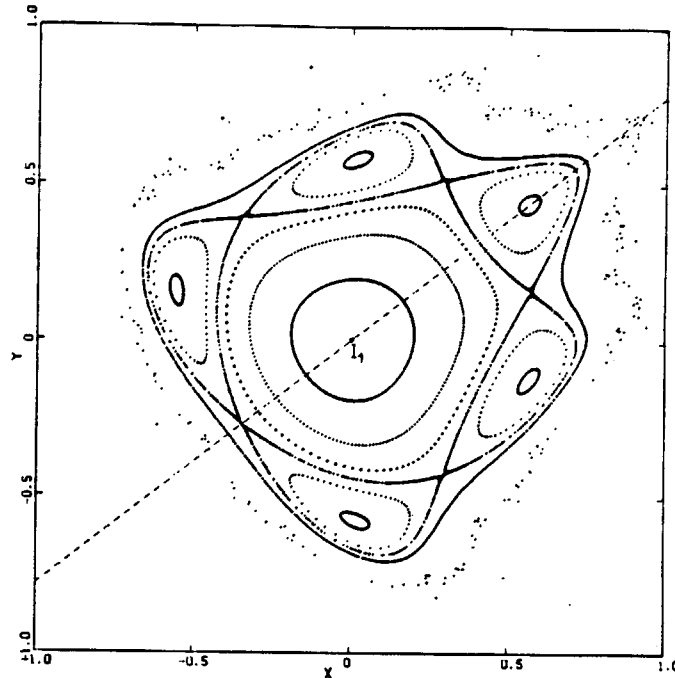


Figure 4.12: *The phase portrait for Henon map with  $A = 1.3284305$ . This picture is directly taken from Henon's paper (Henon 1969). Compare this with the one that KAM finds.*

5. **Missing Islands Rule.** Pick a midpoint (0.05 0.56) between orbit-1 and orbit-4. Result: orbit-5 (5-ISLAND CHAIN).
6. **Missing Separatrix Rule.** Pick two initial states: first, a midpoint (0.298 0.519) between two islands of orbit-5; second, a midpoint (-0.02 0.48) between orbit-4 and orbit-5. The first initial state results in an orbit which KAM fails to recognize. The second initial state results in orbit-6 (5-ISLAND CHAIN).
7. **Missing Separatrix Rule.** Pick two initial states: first, a midpoint (0.299 0.519) between two islands of orbit-6; second, a midpoint (0.39 -0.24) between orbit-4 and orbit-6. The first initial state results in orbit-7 (5-loop SEPARATRIX). The second initial state results in orbit-8 (5-ISLAND CHAIN).
8. **Missing Islands Rule.** Handle complaint caused by inconsistency between orbit-2 and orbit-3. Pick a midpoint (-0.08, -0.12) between orbit-2 and orbit-3. Result: an orbit which KAM fails to recognize. (KAM tries three different midpoints, but fails to recognize any one of them. The current complaint is therefore removed.)
9. **Boundary Circle Rule.** Orbit-1, the outermost quasiperiodic orbit, is not consistent. Pick some random point (0.86 0.08) outside orbit-1. Result: orbit-9 (ESCAPE ORBIT).

10. **Boundary Circle Rule.** Orbit-1, the outermost quasiperiodic orbit, is not consistent. Pick some midpoint  $(-0.11 \ -0.64)$  between orbit-1 and orbit-9. Result: orbit-10 (QUASIPERIODIC) with  $\rho \in (\frac{4}{21} \ \frac{5}{26})$ .

## 4.7 OTHER EXPERIMENTS

I have tested the consistency maintenance algorithm with more than 30 different values of the parameter  $\alpha \in (0 \ 2.2)$  of the Henon Map. KAM is able to reproduce all phase portraits like those found in Henon's paper. The algorithm has also been successfully run on over 50 phase portraits of two other area-preserving maps: (1) the Bak Map with a cubic nonlinearity, and (2) the Standard Map with a toroidal phase space. In this section, I will single out three experiments that are more complicated than the one shown in Fig. 4.11, and indicate how KAM is extended to handle these cases. It should be mentioned here that the extensions are not *ad hoc*; they are all that are needed for handling a general area-preserving map, and this is a consequence of a theorem that provides a complete classification of generic bifurcations of periodic points (see Chapter 5).

### 4.7.1 Phantom 3-Kiss Bifurcation

In general, KAM does not search a region bounded by two compatible quasiperiodic orbits because island chains are not expected there. There are, however, two exceptional situations<sup>9</sup>: the Phantom 3-Kiss and Phantom 4-kiss bifurcation. I shall illustrate my point by the Phantom 3-Kiss Bifurcation. A similar explanation carries through to the other case.

Consider the phase portrait of the Henon Area-preserving Map with parameter value  $\alpha = 2.0042417$  (i.e.,  $\cos \alpha = -0.42$ ). Fig. 4.14a shows two quasiperiodic orbits. Their rotation numbers are bounded by the same interval  $(\frac{10}{31} \ \frac{1}{3})$ ; therefore, the orbits are compatible. Yet, as we shall see, these two orbits bound a large 3-island chain. Let me explain, again using the incompressible fluid analogy, how the island chain can arise.

---

<sup>9</sup>We know there are *only two* exceptions because, among all the periodic orbits created by the generic bifurcations for area-preserving maps, only those coming out of these two bifurcations have the exceptional geometry. See Chapter 5 on bifurcations.

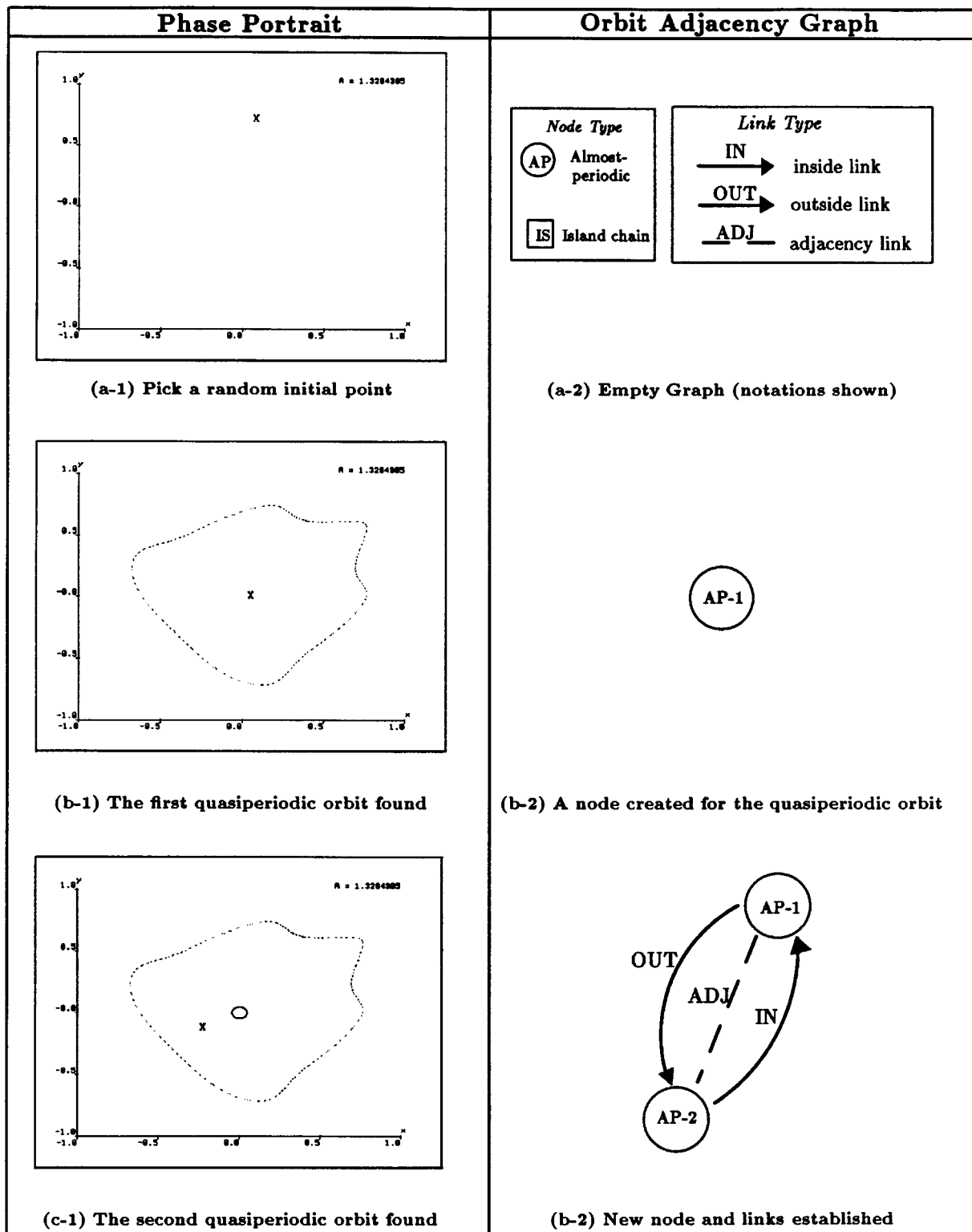


Figure 4.13: How KAM picks initial states. Left: Snapshots of phase portraits. Crosses (×) indicate the new initial states picked. Right: Data structure representing parts of the Orbit Adjacency Graph.

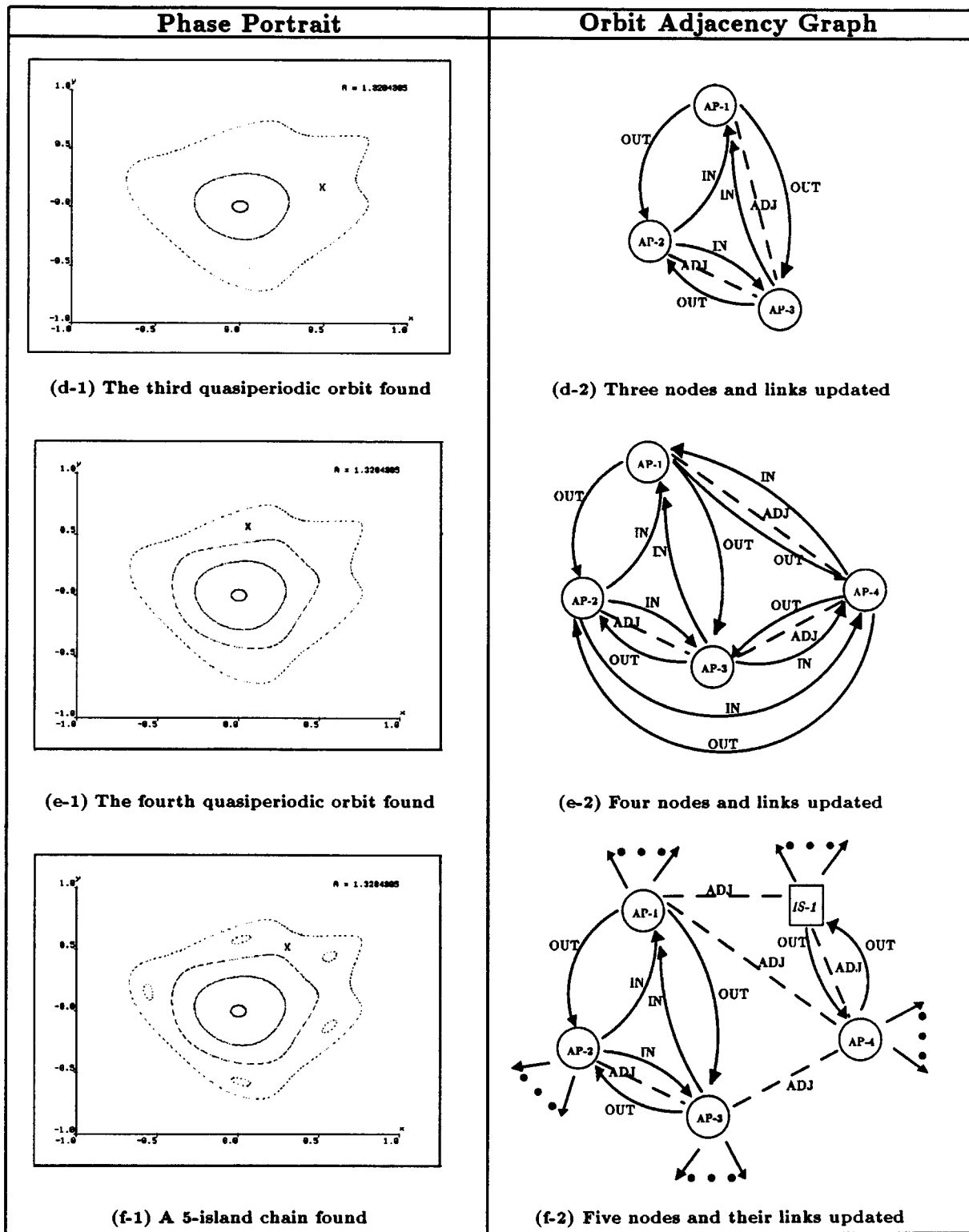


Figure 4.13: (cont.) How KAM picks initial states. Left: Snapshots of phase portraits. Crosses (x) indicate the new initial states picked. Right: Data structure representing parts of the Orbit Adjacency Graph.

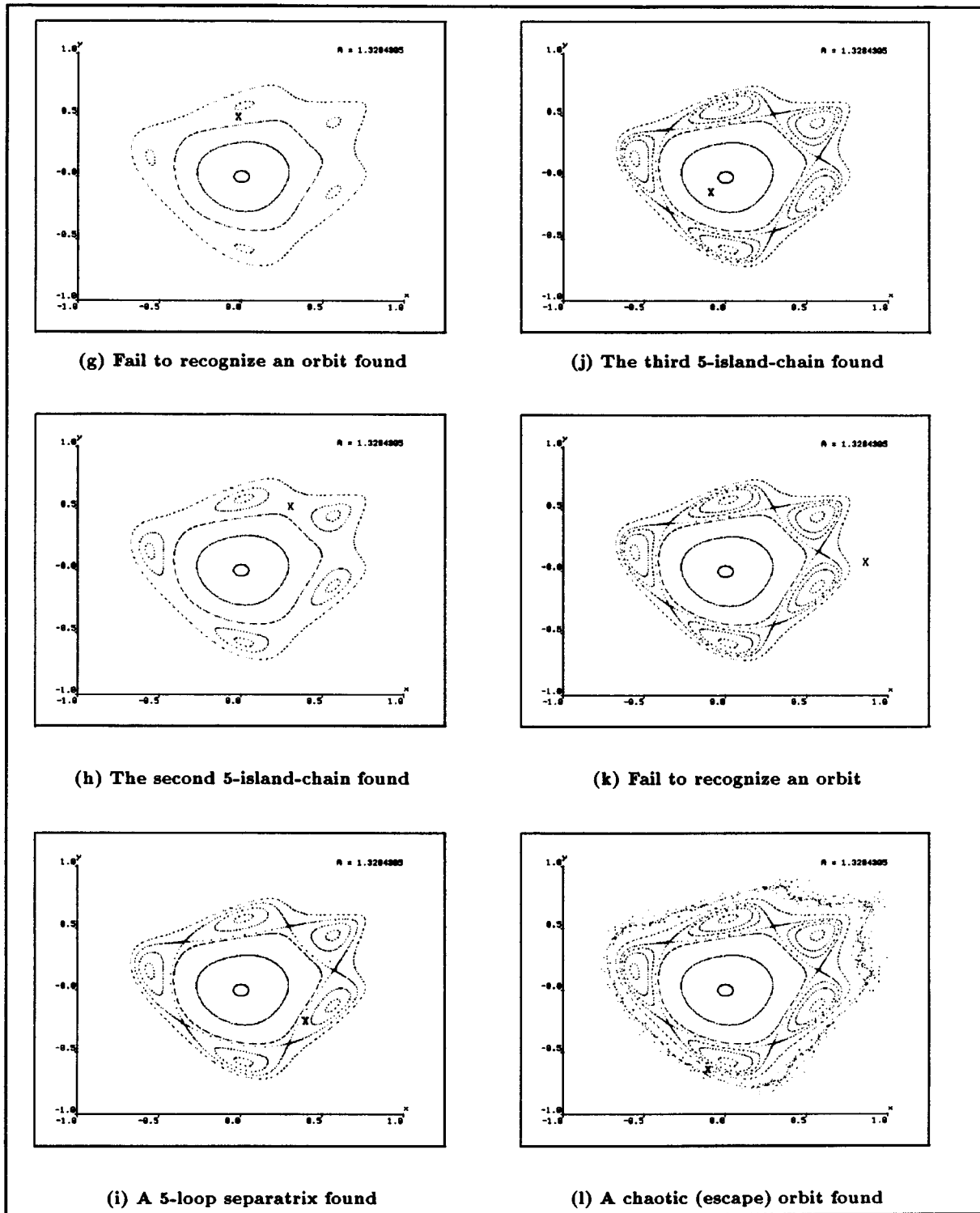
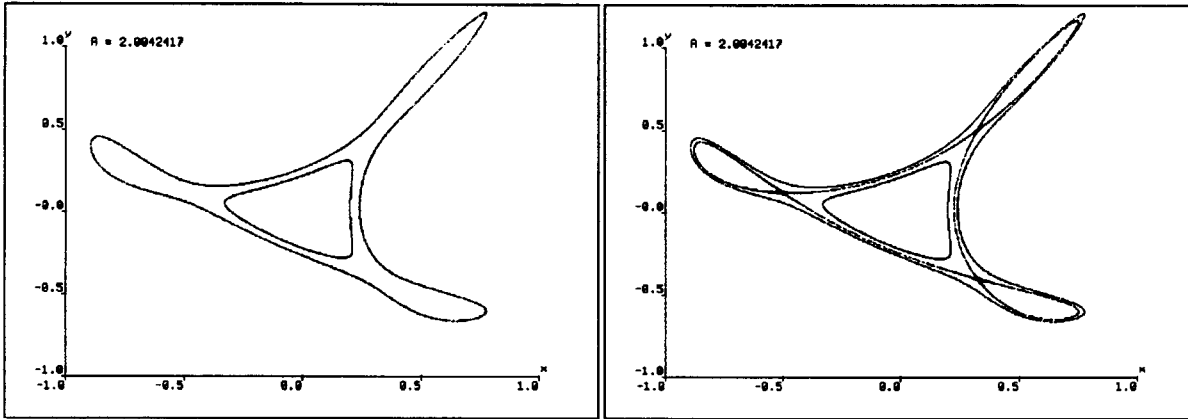
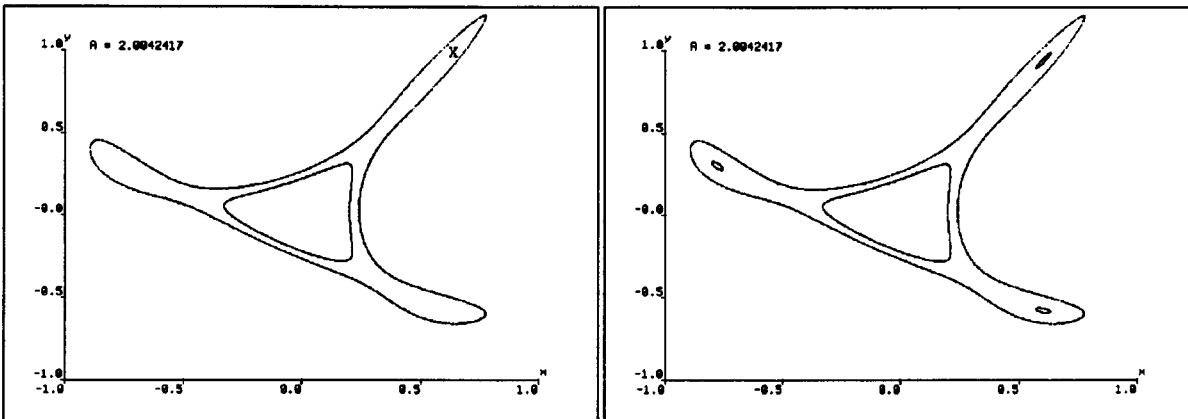


Figure 4.13: (cont.) How KAM picks initial states: the remaining snapshots showing how the rest of the orbits are found.



(a) Two orbits with compatible rotation number

(b) Squeezing the 3-starfish orbit



(c) An initial state picked near a tip of the starfish orbit

(d) A 3-island chain found

Figure 4.14: *Looking for island chains created by Phantom 3-kiss bifurcation.*

Two flow lines are displayed in Fig. 4.14a. The outer one is “starfish-like” with three radially disposed arms about a central disk. The inner one is an ordinary circular flow line.<sup>10</sup> Because both flow lines circulate in the same direction, they are compatible. Imagine squeezing the part of the starfish flow line in the region connecting one of the arms and the central disk (see Fig. 4.14b). Inside this arm, a roller begins to take its

<sup>10</sup>The flow line actually looks triangular in this particular case.

shape. It is not hard to see an elliptic stagnant point resides in the center of the roller.  
11

Since the appearance of starfish-like orbits is a prominent feature of this type of bifurcation, KAM distinguishes starfish orbits from ordinary ellipse-shaped quasiperiodic orbits.

A starfish orbit is a quasiperiodic orbit whose shape consists of either 3 or 4 radially disposed arms about a central disk. The arms can be detected by rapid change in the curvature of the boundary. The length of an arm relative to the radius of the central disk provides a measure of the protrusiveness of the arm. Currently, KAM requires the arm to be at least two times longer than the disk for an orbit to qualify as starfish-like. When a starfish orbit is found, KAM selects a point, inside the orbit, close to the tip of one of the arms. This point is a reasonable approximation to the location of the island chain, if any, enclosed by the orbit. Fig. 4.14c shows a typical point chosen by KAM. The point leads to a 3-island chain in the next trial (Fig. 4.14d).

To detect rapid changes in boundary curvature, 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 coordinate functions,  $x(s)$  and  $y(s)$ , are smoothed by the Gaussian and its first two derivatives at spatial scale of 11. Finally, the zero-crossings of the curvature function  $\kappa(s)$ , and the signs of  $\kappa(s)$  are computed to determine the locations and type of the extrema.

## 4.7.2 Bak Map: Rimmer Bifurcation

In a typical area-preserving map, a chain of islands surrounding a fixed point belongs to a single orbit. However, in maps with additional symmetry, such as the Bak Map, the islands can be made up by two island chains each consisting of half the islands. A good illustration is shown in the phase portrait of the Bak Map with parameter value  $A = 2.0$  (see Fig. 4.15). The ten islands, on the border of the regular region, belong to two alternating 5-island chains.

This type of bifurcation – known as **Rimmer bifurcation**, or **double m-bifurcation** – gives rise to twice as many periodic orbits as the typical bifurcation. Double  $m$ -bifurcation, though non-generic within the class of area-preserving maps, can be shown

---

<sup>11</sup>That a 3-island chain is frequently found inside a starfish-like orbit is a consequence of extremal bifurcation (see Chapter 5).



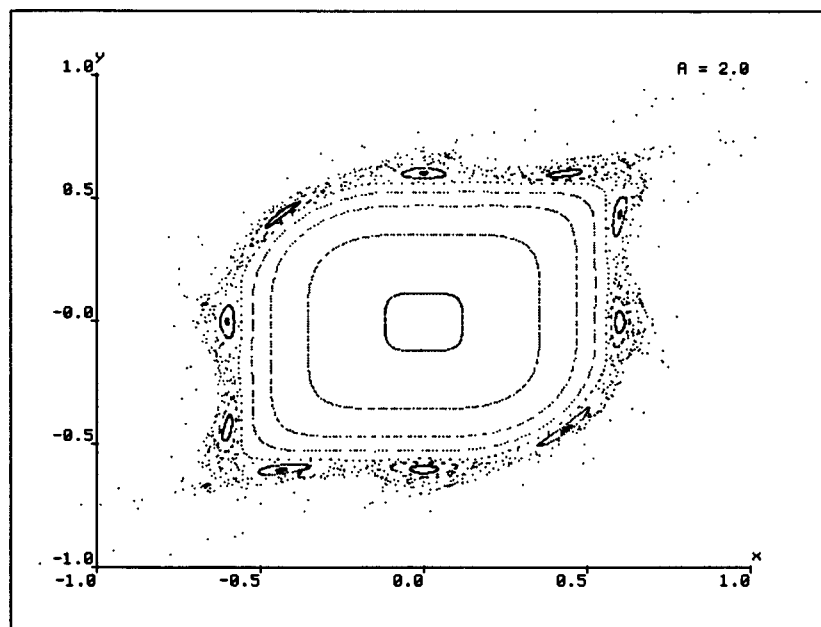


Figure 4.15: *The phase portrait of Bak Map with  $A = 2.0$ .*

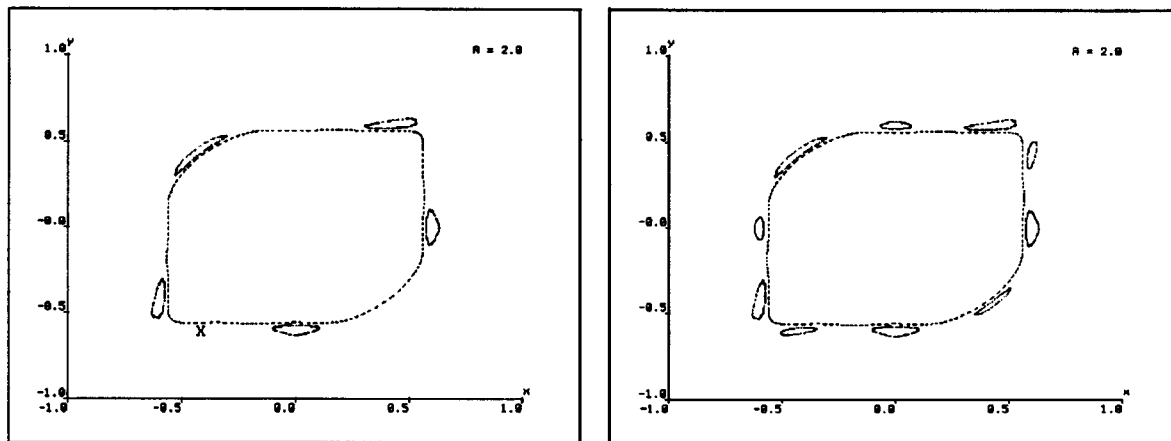
to be generic for reversible maps [26].<sup>12</sup> It is observed [16] that the fixed point of a reversible map has normal  $m$ -bifurcation when  $m$  is even, but double  $m$ -bifurcation when  $m$  is odd.

The double  $m$ -bifurcation creates some difficulty for phase space searching. Finding the first island chain around the fixed point is routine. However, once the first island chain is found, the Orbit Adjacency Graph is pairwise consistent in the neighborhood of this island chain; KAM does not know it has to look for the second of the pair of island chains created by the double  $m$ -bifurcation. So, KAM needs to recognize a potential double  $m$ -bifurcation, and search for an island chain pair.

Notice the arrangement of the two island chains in Fig. 4.15. The two chains are rotationally symmetric – their islands are alternating. A heuristic making use of this symmetry is built into the Quasiperiodic-and-Island suggestion rule (see section 4.4.3)

<sup>12</sup>A map  $S$  is called a reflection if  $S \circ S = \text{Identity}$  and the Jacobian determinant of  $S$  is negative. We say a map  $T$  is reversible if  $T$  is invertible and  $T^{-1} = S \circ T \circ S$ . In words, a reversible map has the property that the reflection (by  $S$ ) of the image (under  $T$ ) of a point  $p$  is the inverse image of  $p$ 's reflection. All area-preserving maps derivable from hamiltonians even in momenta are reversible. However, not all area-preserving maps are reversible.

specifically to look for the symmetric daughter of an  $m$ -island chain whenever  $m$  is odd. Fig. 4.16a shows the candidate initial state proposed by the heuristic once the 5-island chain is found. The sister 5-island chain is found in the next trial (Fig. 4.16b). The heuristic is effective in handling all the double  $m$ -bifurcations arising from the Bak Map and the Standard Map.



(a) An example of Rimmer bifurcation.  
(x indicates initial state proposed.)

(b) A second 5-island chain found

Figure 4.16: *Looking for Rimmer island chains*

### 4.7.3 Standard Map: Toroidal Phase Space

The Consistency Maintenance Algorithm is also tested on the Standard Map, whose phase space is a torus. The torus is not topologically equivalent to the euclidean plane. For instance, if one draws any closed curve on the euclidean plane, the curve bounds a region on the surface. This is not true on the torus: there exists some closed curve on the torus that does not bound any area (see Fig. 4.17). Consequently, the algorithm to determine orbit adjacency has to be modified to handle the contour propagation of a non-bounding closed curve. Certain suggestion rules that refer to the boundary of the phase space are also changed because, unlike a compact subset of the euclidean plane, the torus does not have a boundary. For instance, when there are no orbits outside a boundary circle, the **boundary circle suggestion rule** will suggest a midpoint between a point on the orbit and the point farthest from it on the orbit with the distance being measured along the exterior of the orbit.

The basic consistency algorithm, however, remains intact. A typical result is shown in Fig. 4.18 with the parameter value  $k = 1.4$ . KAM takes about 8 hours of computer time to find 22 representative orbits from a total of 37 trials.

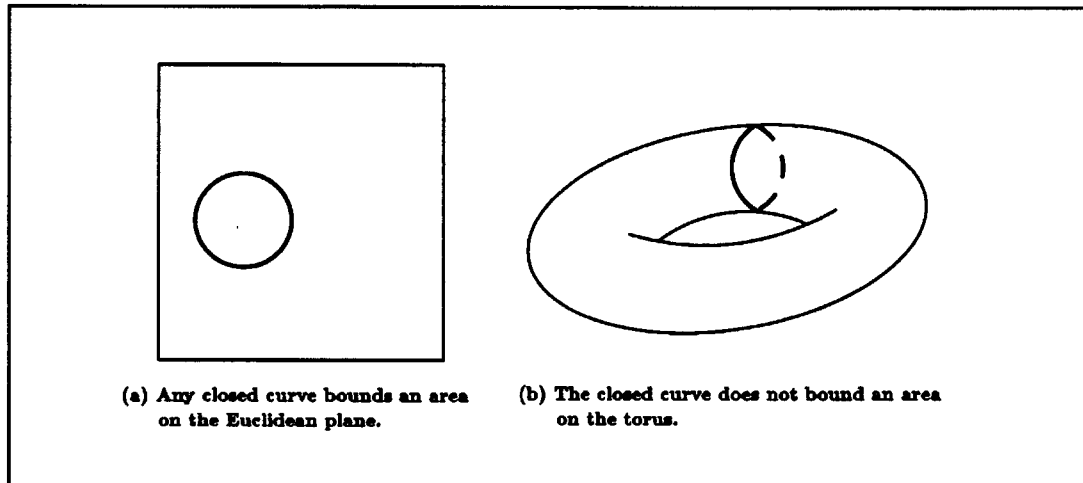
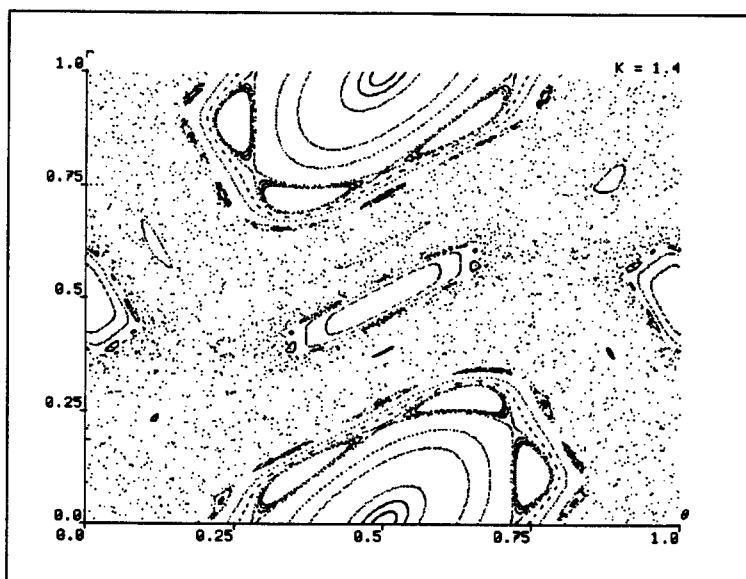


Figure 4.17: *The torus is not topologically equivalent to the Euclidean plane because one can draw non-bounding closed curves on the toroidal surface.*

The phase portrait of the Standard Map is more complicated than that of the Henon Area-preserving Map or the Bak Map. For example, whereas all periodic orbits of the two plane area-preserving maps bifurcate from a single fixed point, the periodic orbits of the Standard Map come from different families. Four such families are prominent in Fig. 4.18. The first family originates from the fixed point at  $(\frac{1}{2}, 0)$ . The local structure around this fixed point is similar to those we have seen before – the fixed point being surrounded by, in this case, a ordinary 6-island chain and two alternating 7-island chains (from Rimmer bifurcation). The second family bifurcates from the period-2 orbit in the center of the phase portrait. Two Rimmer 10-island-chains oscillate about the period-2 orbit. The third and fourth family of periodic orbits consist of the 3-island chains located between the first and the second family.

## 4.8 CONCISE PHASE PORTRAIT DESCRIPTION

We have seen how KAM searches for representative orbits in the phase space by identifying locally compatible orbits. These orbits are aggregated in the orbit adjacency graph, which contains a great deal of information about the structure of the phase portrait. Our



The phase portrait has 4 canonical flow patterns.

\*\*\*\*

**FIRST CANONICAL FLOW PATTERN:**

The portrait has an elliptic fixed point at (0.5 1.0).

Surrounding the fixed point is a regular region bounded by a KAM curve with rotation number between  $1/6$  and  $6/35$ .

Outside the regular region lies a chain of 6 islands.

The island chain is bounded by a second KAM curve with rotation number between  $3/20$  and  $2/13$ .

Outside the second KAM curve are two Rimmer 7-island-chains.

The Rimmer island chains are surrounded by CHAOTIC orbits.

\*\*\*\*

**SECOND CANONICAL FLOW PATTERN:**

The portrait has elliptic period-2 periodic points at ((0.5 0.5) (0.0 0.5)).

Surrounding the primary 2-ISLAND-CHAIN are two Rimmer 10-ISLAND-CHAINS.

Outside the regular region lie CHAOTIC orbits.

Chaotic orbits connect this canonical flow pattern with neighboring flow patterns.

\*\*\*\*

**THIRD CANONICAL FLOW PATTERN:**

The portrait has elliptic period-3 periodic points at ((0.881 0.769) (0.5 0.62) (0.119 0.62)).

Outside the regular region lie CHAOTIC orbits.

Chaotic orbits connect this canonical flow pattern with neighboring flow patterns.

\*\*\*\*

**FOURTH CANONICAL FLOW PATTERN:**

The portrait has elliptic period-3 periodic points at ((0.883 0.383) (0.117 0.234) (0.5 0.383)).

Outside the regular region lie CHAOTIC orbits.

Chaotic orbits connect this canonical flow pattern with neighboring flow patterns.

**GLOBAL CHAOS APPEARS:** chaotic orbits occupy 71% of phase space.

Figure 4.18: The phase portrait of Standard Map with  $K = 1.4$ . The toroidal phase space is represented by the "flat" torus. The torus is cut along two circles – one vertical and another horizontal – and flattened into a unit square. The symbolic description of the phase portrait is displayed in the lower portion of the figure.

goal in this section is to show how KAM generates a concise qualitative description of the phase portrait from the orbit adjacency graph. Just as we use the notion of orbit to compactly describe and make explicit the inherent structure of a large collection of points in the phase space, we introduce another concept, the **canonical flow pattern**, to represent the structure of an aggregate of orbits.

A canonical flow pattern consists of three types of orbits: (1) a **primary orbit**, (2) quasiperiodic orbits surrounding the primary orbit, and (3) **secondary island chains** bifurcating from the primary orbit. Informally, the primary orbits are stable fixed points or stable periodic orbits that exist in the limit of zero perturbation strength. For example, the origin  $(0,0)$  of the Henon Map (as in Fig. 4.11) is a primary fixed point because it exists even when the perturbation term,  $x^2$ , vanishes. Another example is the stable period-2 orbit at  $(\frac{1}{2}, \frac{1}{2})$  and  $(0, \frac{1}{2})$  of the Standard Map in Fig. 4.18. This period-2 orbit exists in the limit the perturbation parameter  $K \rightarrow 0$ . Island chains that are not primary are called secondary island chains.

Each canonical flow pattern characterizes the qualitative features of a portion of the phase portrait by describing the geometry of orbits near a primary orbit. Since multiple primary orbits are common, a typical phase portrait consists of several canonical flow patterns. The phase portrait in Fig. 4.18, for instance, has 4 primary orbits – a fixed point at  $(\frac{1}{2}, 0)$ , the period-2 orbit just mentioned, and two period-3 orbits. Each primary orbits and the secondary orbits associated with it constitute one canonical flow pattern. We say two canonical flow patterns are *connected* if there exist some orbit that can move freely between the neighborhoods of the two flow patterns. To take an example, the four canonical flow patterns in Fig. 4.18 are all connected.

Partitioning the orbit adjacency graph into canonical flow patterns involves 5 steps:

1. Identifying primary orbits
2. Grouping orbits
3. Simplifying orbit structures
4. Classifying canonical flow patterns
5. Determining connectivity of canonical flow patterns

### 4.8.1 Identifying Primary Orbits

Classifying the geometry of the orbits allows simple visual identification of the primary orbits. Specifically, the primary orbits are elliptic fixed points, or elliptic periodic points that do not “enclose” periodic points of smaller periodicity. According to this criterion, all elliptic fixed points are primary orbits. Periodic orbits, however, can be either primary or secondary. For instance, consider the period-6 periodic points inside the 6-island chains shown in Fig. 4.18. We see that these period-6 periodic points enclose the fixed point (or period-1 periodic point) at  $(\frac{1}{2}, 0)$ , and so they cannot be a primary orbit.<sup>13</sup> Another example is the period-2 periodic points inside the 2-island chains in the middle portion of Fig. 4.18. This time these period-2 periodic points do not enclose the fixed point at  $(\frac{1}{2}, 0)$ . Since  $(\frac{1}{2}, 0)$  is the only fixed point of this phase portrait, we can conclude that the period-2 periodic points form a primary orbit. Similarly, the two period-3 orbits are primary orbits because they do not enclose any fixed point or period-2 orbits.

The procedure to identify primary orbits consists of two steps. First, collect all elliptic fixed points and elliptic periodic orbits. Second, remove periodic orbits that enclose periodic orbits of smaller periodicity.

How does the program determine when one periodic orbit encloses another? Suppose we have two periodic orbits A and B. Further suppose A is period  $m$ , and B is period  $n$ . Let us also assume  $m > n$ . If A encloses B, then each point of B has to be surrounded by an equal number of points from A. So, A cannot enclose B unless  $m$  is an integer multiple of  $n$ . Let us say  $m = k \times n$  for some positive integer  $k$ . To determine if A encloses B, the following simple heuristic is used. For each point  $p_B$  of orbit B, find its  $k$  closest points from orbit A. Let  $A_c$  be the centroid of these  $k$  points, and  $R_A$  be the average distance of the  $k$  points from  $A_c$ . Construct a circle centered at  $A_c$  with radius  $= f \times R_A$  for some small positive factor  $f$ .<sup>14</sup> We say A encloses B if each  $p_B$  of orbit B lies inside the corresponding circle. See Fig. 4.19.

### 4.8.2 Grouping Orbits

Orbits that are not primary are called **secondary**. After the primary orbits are identified, the next step is to partition secondary orbits into groups surrounding the primary orbits.

---

<sup>13</sup>Recall that the phase space of the Standard Map is a torus. The upper and lower edges are identified, and so are the left and right edges.

<sup>14</sup>In the current implementation,  $f$  is arbitrarily chosen to be 0.15. There is nothing special about 0.15; any number slightly greater 0 seems to work as well.

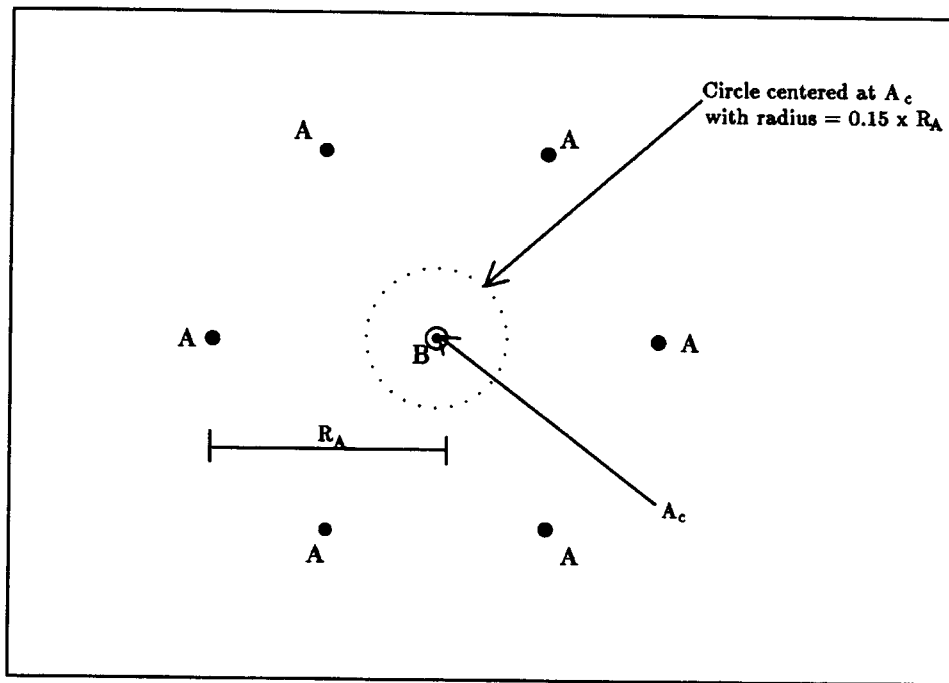
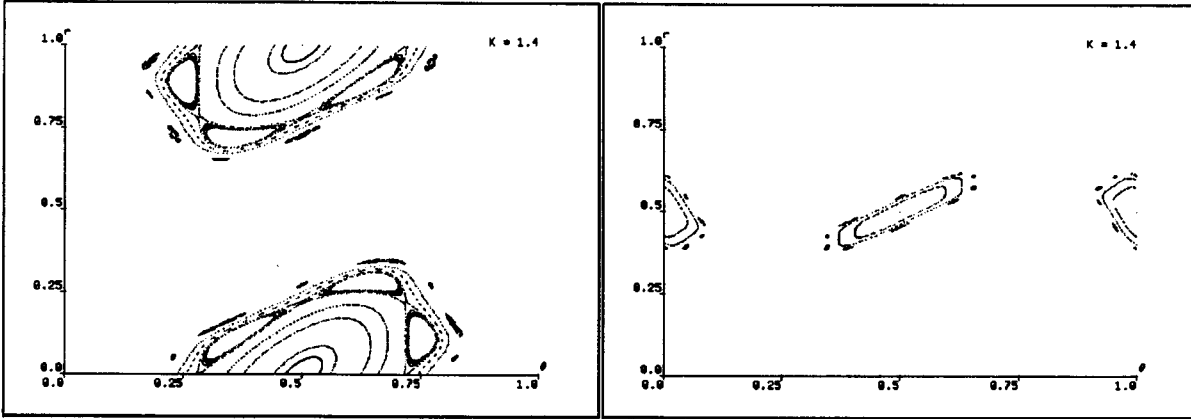


Figure 4.19: The period-6 orbit  $A$  (indicated by  $\bullet$ ) encloses the fixed point  $B$  (indicated by  $\circ$ ) because  $B$  lies within the circle (shown as a dotted circle), centered at the centroid of orbit  $A$ , with radius  $= 0.15 \times R_A$  where  $R_A$  is the average distance of a periodic point from the centroid. In this example, we have  $m = 6$ ,  $n = 1$ , and  $k = 6$ .

The criterion for associating a secondary orbit  $S$  with a primary orbit  $P$  depends on the type of  $S$ . Specifically, four rules are used:

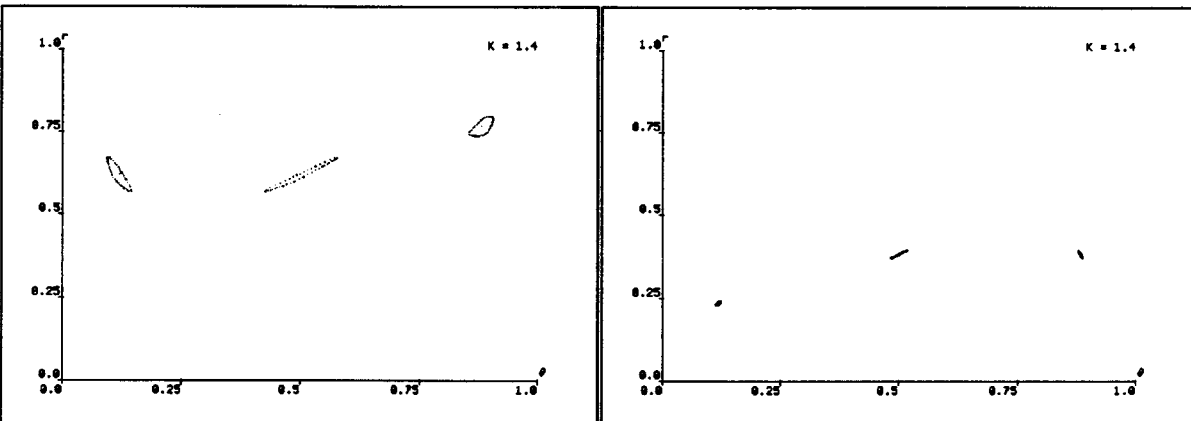
1.  $S$  is a bounding quasiperiodic orbit: Associate  $S$  with  $P$  if  $P$  is in the inside of  $S$ .
2.  $S$  is an island chain: Associate  $S$  with  $P$  if the periodic orbit inside  $S$  encloses  $P$ .
3.  $S$  is a separatrix: Associate  $S$  with  $P$  if some secondary island chain of  $P$  is inside  $S$ .
4.  $S$  is a chaotic, an escape, or a non-bounding quasiperiodic orbit: Associate  $S$  with  $P$  if  $S$  is adjacent to some secondary orbit of  $P$ .

The results of grouping orbits for Standard Map with  $K = 1.4$  are shown in Fig. 4.20.



(a) Secondary orbits around the fixed point

(b) Secondary orbits around the period-2 orbit



(c) A 3-island chain around the top period-3 orbit

(d) A 3-island chain around the bottom period-3 orbit

Figure 4.20: Grouping orbits around primary orbits in the Standard Map phase portrait with  $K = 1.4$ . Chaotic orbits are not shown.

### 4.8.3 Simplifying Orbit Structures

In general, not all secondary orbits are essential for the qualitative description of the flow pattern near their associated primary orbit. For example, the secondary orbits shown in Fig. 4.20a contain redundant nested 6-island chains. Of the three nested 6-island chains, the largest one gives the best estimate of the phase space region enclosed by these island



chains. We say such an island chain is **maximal** because it is the outermost of a sequence of nested island chains having the same period. Similarly, the four nested non-maximal quasiperiodic orbits enclosing the fixed point are also redundant.

The third step is therefore to simplify the group of orbits around a primary orbit by removing redundant orbits. Three types of orbits are removed: (1) non-maximal island chains, (2) non-maximal quasiperiodic orbits, and (3) high-order island chains. An island chain with a large number of islands usually occupies an insignificant phase space area, and so it has little effects on the overall dynamics. Fig. 4.21 shows the remaining five secondary orbits around the fixed point after nine redundant orbits are removed.

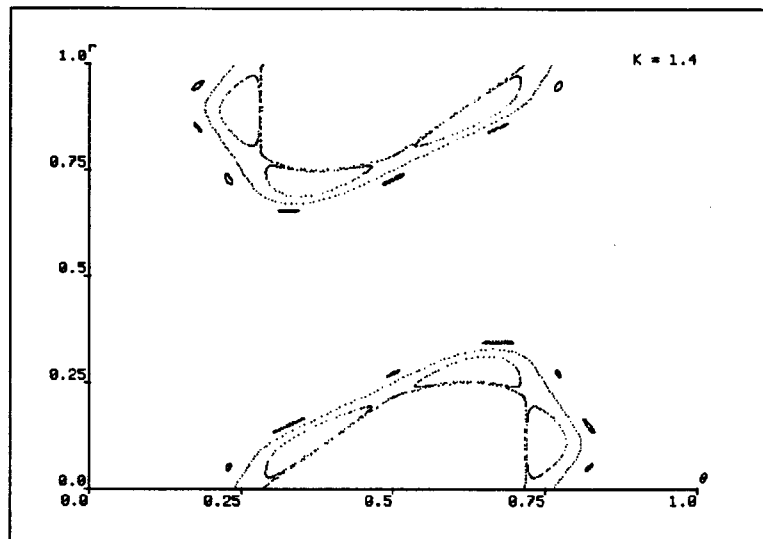


Figure 4.21: *Removing nine redundant orbits around the fixed point at  $(\frac{1}{2}, 0)$  results in five essential secondary orbits that capture the structure of the flow pattern around the fixed point.*

#### 4.8.4 Classifying Canonical Flow Patterns

Simplifying the group of secondary orbits around a primary orbit generates what I call a **canonical flow pattern** – the characteristic arrangement of island chains and KAM curves around a primary orbit. The qualitative description of the phase portrait consists of a geometric description of the orbits comprising the canonical flow pattern, and how the canonical flow patterns are related. Thus it is natural to ask when two canonical

flow patterns have the same description. The fourth step, therefore, is to assign semantic labels to the canonical flow patterns.

Classifying canonical flow patterns begins by placing an equivalence relation on the set of canonical flow patterns. A good classification requires a natural equivalence relation, and a set of computable invariants that identify an individual equivalence class. An equivalence relation should be *natural* in the sense that it is based on qualitative resemblances of the orbits structures of the phase portraits. On the other hand, the equivalence relation should also be *tractable* in the sense that there exist computable numerical or algebraic invariants that are equal for canonical flow patterns belonging to the same equivalence class, and are different if the canonical flow patterns belong to different equivalence classes.

An equivalence relation fundamental in dynamical systems theory is **topological conjugacy**. Two discrete maps  $f$  and  $g$  are topologically conjugate if there exists a homeomorphism  $h$  such that  $h \circ f = g \circ h$

The intuition behind the definition is the idea that two geometrical objects – phase portraits, in this case – are the same if there is a certain type of change of coordinates that transforms one object into the other. A circle, for example, under a linear change of coordinates, can be transformed into an ellipse, and hence they are qualitatively similar. Of course, how fine the classification is depends on the restriction on the allowable type of coordinate changes. Equivalence under a continuous change of coordinates – a homeomorphism – corresponds to a topological conjugacy; equivalence under a smooth change of coordinates (i.e., the transformation has continuous derivatives), a *smooth conjugacy*. Depending on the particular problem, we may desire a finer or coarser classification of equivalent behaviors.

One reason why topological conjugacy is commonly used is that discrete maps that are intuitively non-equivalent are in fact classified as different equivalence classes. Although topological conjugacy is natural, it seems to be too strong for our purpose. To see this, we first note that topological conjugacy preserves three important invariants – fixed points, periodic orbits, and rotation number. So, if two maps differ in any of these invariants, then they are not topologically conjugate. For example, orbits with rotation number  $\frac{1}{5}$  and  $\frac{2}{5}$  are not topologically conjugate even if they are enclosed by island chains with identical appearance. Another problem is that topological conjugacy does not take into account the relative importance – measured in terms of phase space area occupied by the enclosing island chain – of periodic orbits. Thus, if two phase portraits differ by even one small island chain, they are still not topologically conjugate.

Because topological conjugacy classifies intuitively similar phase portraits as non-equivalent, and because periodic orbits associated with small island chains are difficult to find numerically, we are led to consider a weaker equivalence relation that ignores details of rotation number and insignificant island chains. In the rest of this section, I will introduce a classification based on a notion of qualitative equivalence, which has not been studied in the dynamical systems literature.

Canonical flow patterns fall into two main categories: (1) F-FLOW-PATTERN, and (2) P-FLOW-PATTERN. The prefix letter “F” or “P” identifies the type of the primary orbit. A F-FLOW-PATTERN refers to a collection of orbits surrounding a fixed point. Similarly, a P-FLOW-PATTERN designates orbits surrounding a periodic orbit. Thus, for example, the canonical flow pattern in Fig. 4.21 is a F-FLOW-PATTERN, whereas those in Fig. 4.20b,c,d are P-FLOW-PATTERNS.

Within each category of flow pattern, the canonical flow pattern is further characterized by four indices  $m$ ,  $n$ ,  $p$  and  $q$ . The convention is as follows:

1.  $m$  = number of (ordinary) island chains
2.  $n$  = number of (ordinary) island chains that are bounded by a KAM curve
3.  $p$  = number of Rimmer island chain pairs
4.  $q$  = number of Rimmer island chain pairs that are bounded by a KAM curve

I use the notation “F-FLOW-PATTERN- $m$ - $n$ - $p$ - $q$ ” or “P-FLOW-PATTERN- $m$ - $n$ - $p$ - $q$ ” to identify the type of a canonical flow pattern. Also for convenience, when  $p = q = 0$ , I use the shorthand “F-FLOW-PATTERN- $m$ - $n$ ” or “P-FLOW-PATTERN- $m$ - $n$ ”.

For an example of categorization, consider again the F-FLOW-PATTERN in Fig. 4.21. Since the 6-island chain around the fixed point is the only non-Rimmer island chain, and it is bounded by a KAM curve, we have  $m = n = 1$ . Outside the 6-island chain, there is a pair of Rimmer 7-island chains, but these island chains are not bounded by any KAM curves. So we have  $p = 1$ , and  $q = 0$ . The type of this canonical flow pattern is written as F-FLOW-PATTERN-1-1-1-0.

A second example is the flow pattern shown in Fig. 4.20b. Here the primary orbit is a period-2 orbit. Since the primary orbit is surrounded by a pair of Rimmer 10-island chains which are not bounded by any KAM curves, we have  $m = n = 0$ ,  $p = 1$ ,

and  $q = 0$ . The type of canonical flow pattern is written as P-FLOW-PATTERN-0-0-1-0. Note: the 2-island chain enclosing the period-2 orbit is *not* considered an island chain in the classification because each of the islands is thought of as a quasiperiodic orbit surrounding one of the period-2 points when we iterate the square of the map. This convention gives consistent meanings to the four indices in both the F-FLOW-PATTERN and the P-FLOW-PATTERN.

In a similar manner, we classify the remaining two flow patterns in Fig. 4.20c,d as P-FLOW-PATTERN-0-0 because the period-3 orbits are not surrounded by any island chain.

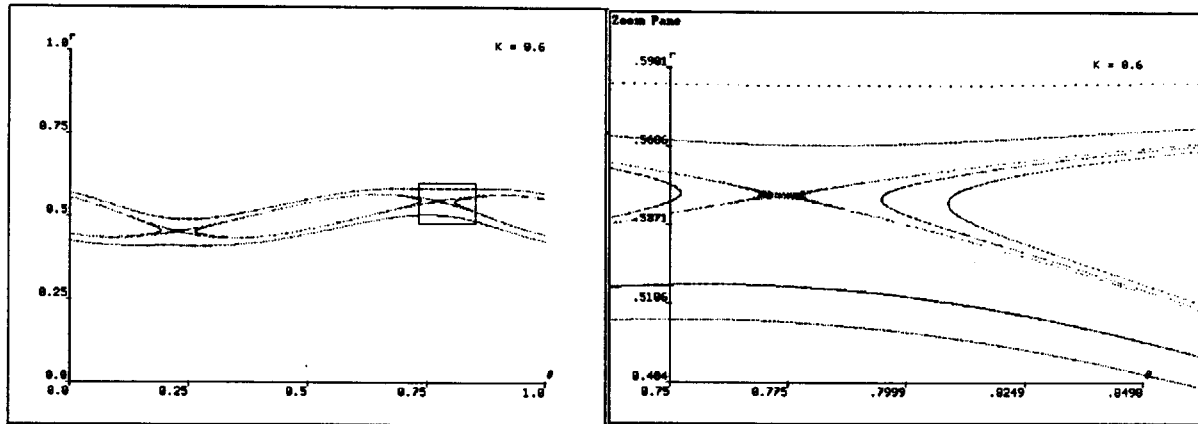
Finally, we define the notion of qualitative equivalence of canonical flow patterns:

**Definition 5** *Two canonical flow patterns are qualitatively equivalent if (1) they have the same primary orbit, (2) they have identical indices, and (3) their secondary island chains have the same spatial relationships with respect to the primary orbit.*

The definition of qualitative equivalence implies two properties. First, the equivalence relation does not preserve rotation numbers. Second, the equivalence relation by itself does not ignore insignificant island chains. Small island chains are instead removed in the third step, the simplification of orbit structures.

#### 4.8.5 Determining Connectivity of Canonical Flow Patterns

In section 3.1, we saw that motion near a homoclinic orbit (another name for a separatrix) is quite complicated – points on the orbit are extremely sensitive to small disturbances. In fact, the orbit is proven to be chaotic; its points hop from place to place in a random manner. For a small nonlinearity, these separatrices are thin because they are bounded on either side by KAM curves (Fig. 4.22). So these homoclinic orbits are isolated from each other. As the nonlinearity increases, these KAM curves separating regions of chaotic motion are distorted and eventually destroyed. The chaotic regions from neighboring separatrices thus begin to merge. The merging of chaotic zones subsequently leads to the appearance of **global or connected chaos**. The transition from local or isolated chaos to global chaos is an important event because it signifies a sudden increase in the relative phase space area covered by chaotic motion.



(a) A thin chaotic layer bounded by KAM curves      (b) Magnifying chaotic layer near separatrix

Figure 4.22: *Local or isolated chaos illustrated by the Standard Map*

Because appearance of global chaos is a global property of phase portrait, a canonical flow pattern, which describes the orbit structures (locally) in some neighborhood of a stable fixed point or stable periodic orbit, cannot by itself reveal the relevant information. However, by examining how the canonical flow patterns are related to each other, one can recognize the appearance of global chaos. In particular, if all canonical flow patterns are connected – i.e., there exist some (chaotic) orbits that can move freely among the neighborhoods of the canonical flow patterns – then barrier orbits, KAM curves that isolate thin chaotic layers, can no longer exist. So, the connectivity of canonical flow patterns give us a simple, geometric way of recognizing global chaos (see the bottom panel of Fig. 4.18).

## 4.9 EVALUATING THE PERFORMANCE

### 4.9.1 Missing Small Structures

The phase portrait for an area-preserving map is considerably more complicated than that apparent from Fig. 4.18. First of all, the Poincaré-Birkhoff theorem [14] tells us that typically there are alternating chains of elliptic and hyperbolic periodic orbits corresponding to *every* rational rotational number (see section 5.1 and 5.2 for more detail). Secondly, the island structures of an area-preserving map are self-similar: the geometric

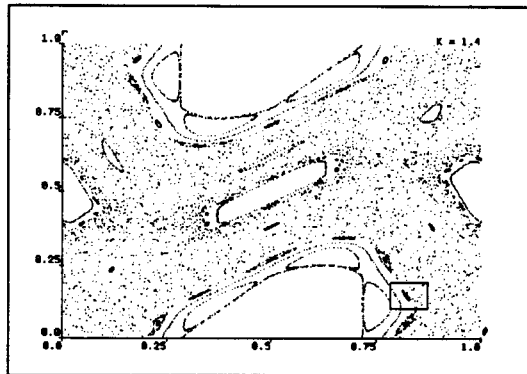
structure of islands surrounding fixed point or periodic point at one length scale is repeated at another length scale. To get a glimpse of this (infinite) complexity, we magnify a small portion of the phase portrait of the Standard Map with  $K = 1.4$  (Fig. 4.23a). Second order island chains are seen to exist near the original large island structures. Further magnifying one island in Fig. 4.23b reveals a third order island structure as seen in Fig. 4.23c. This hierarchy of island structure exists in principle down to the smallest length scale.

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.

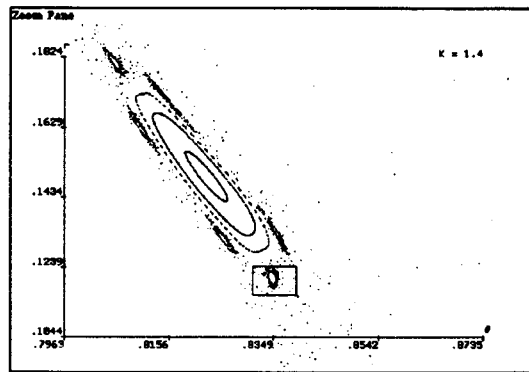
## 4.9.2 What Works Well

When a phase portrait is largely made up of “interesting” orbits, such as quasiperiodic orbits, island chains, or separatrices, KAM matches a domain expert in its ability to select “promising” initial states. This success can be attributed to the geometric representation of the orbits. When two neighboring orbits are inconsistent, the geometry of these orbits delimits a bounded region in the phase space for future search. The boundedness of the region allows the “bisection search” strategy to rapidly zero in on the desired orbits. For instance, the region between two inconsistent quasiperiodic orbits is where island chains can be found. Once an island chain is located, KAM tests the midpoints between consecutive islands to look for hyperbolic periodic points and separatrix. These midpoints become better estimates for hyperbolic points as larger islands are found because the region between consecutive islands shrinks. We have already seen how KAM successfully finds the thin separatrix for Henon Map with parameter  $\alpha = 1.3284305$  in the eighth trial (Fig. 4.13). Such thin separatrices are of course notoriously difficult to locate by random selection of initial states.

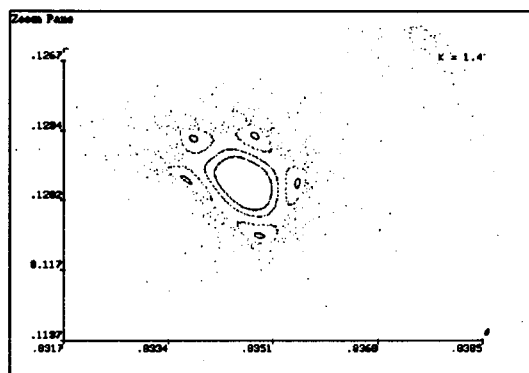
The bisection search strategy is found to be effective for many different phase portraits. In fact, in most cases where KAM fails to locate a separatrix, it is not because KAM cannot find the correct initial state. Instead, it is because it takes too many iter-



(a) Magnifying the boxed region



(b) Revealing second order island structures



(c) Magnifying again reveals third order island structures

Figure 4.23: *Self-similarity in island structures.*

ates <sup>15</sup> to reveal the structure of the separatrix, thereby causing the Orbit Recognition Module to fail.

### 4.9.3 What Does Not Work As Well

The power of the KAM's strategy for locating representative orbits lies in the pairwise orbit consistency rules. Since a chaotic (or escape) orbit imposes no constraint on its neighboring orbits, it stands to reason that KAM will have trouble in regions of phase space largely occupied by chaotic (or escape) orbits. For instance, island chains that are embedded inside a chaotic (or escape) region are difficult to find. KAM handles this problem in a rather *ad hoc* way by looking for large "holes" - collections of neighboring grid cells that are not occupied - inside a chaotic region. This fix is not adequate because low-order island chains may occur in small unoccupied regions. Making smaller grid cells will solve this problem but only at the expense of larger overhead in updating grid status, and determining orbit adjacency. A domain expert has less trouble with this problem mainly because he can recognize the internal shape of a chaotic region. KAM, in contrast, cannot describe the number of holes and connected components that a given two-dimensional dot pattern may have. Extending KAM's shape-recognition ability is a good area for further research.

## 4.10 CONCLUSION: THESIS RESTATED

This chapter shows how the potentially infinite search problem of finding representative orbits in a two-dimensional phase space can be effectively reduced to a simple, local consistency analysis of neighboring orbits. Instead of randomly trying many initial states, KAM carefully examines each orbit found to see if the orbit is compatible with its neighbors. The compatibility of orbits is derived from one basic constraint - the continuity of discrete flow. A simple grammar of behavior, expressed in terms of six production rules, embodies knowledge of primitive orbits, and their modes of transition. KAM focuses its search in regions of phase space where the geometry of orbits violates the grammar. That KAM is able to quickly locate interesting orbit structures, such as separatrices and island chains, is the outcome of KAM's use of a general bisection search strategy to locate missing orbits within the phase space region bounded by the offending orbits.

---

<sup>15</sup>Sometimes, well over several thousand iterates are needed.



This chapter also shows how the **aggregate-partition-classify** paradigm clearly manifests the computational structure of the phase space searching problem. In the first step, orbits are aggregated based on domain-independent, geometric knowledge about spatial adjacency, and inside/outside relations. New orbits are added to the aggregate until all orbits are pairwise consistent. The second step concerns the partitioning of the aggregate of orbits into meaningful groupings. The partitioning requires a mixture of domain-independent, geometric knowledge (for example, when one orbit encloses another), and domain-specific knowledge of dynamics (for example, about the types of primary orbits, and the structure of secondary orbits). The final step is the assignment of semantic labels to the various groupings of orbits. Classifying the groupings is a purely domain-specific task, depending on a notion of qualitative equivalence, which is weaker than topological conjugacy but which is more tractable in the sense that numerical invariants (such as the number of secondary island chains, Rimmer island chains, etc.) can be directly computed.

The power of the consistency maintenance algorithm, and the usefulness of the classification of phase portrait have been demonstrated by running over a hundred experiments involving three different area-preserving maps. In each case, KAM is able to automatically generate phase portraits and their qualitative descriptions that are essentially the same as those appearing in published papers in the experimental dynamics literature. The key insight obtained by these experiments can be stated as this: *Representing behavior geometrically makes possible efficient automatic control of numerical experiments, and high-level, qualitative interpretations of numerical data.* I expect this method of analysis – defining the primitive categories, and cataloging possible transitions among them – pioneered by Waltz in the early seventies, and the emphasis on geometric structures to be a powerful methodology for building intelligent problem-solving partners for scientists and engineers interested in scientific computing.

## Chapter 4 Summary

The phase portrait is an effective tool for representing qualitative dynamics because entire solutions for many initial states are presented simultaneously in a visually graspable format. Phase space searching is the problem of (1) seeking representative orbits of a phase portrait, and (2) interpreting the search results qualitatively.

Efficient search for representative orbits is implemented by a **local consistency maintenance algorithm**, which maintains compatibility between neighboring orbits. The conditions for orbit compatibility can be derived from one basic constraint – the continuity of discrete incompressible flow. A powerful numerical invariant, the **rotation number**, allows simple implementation of orbit incompatibility conditions in terms of six production rules. When two incompatible orbits are found, the region between the two orbits is searched for missing orbits. Resulting orbits are **aggregated** in the **orbit adjacency graph**, which keeps track of the spatial relationships among orbits. The search process is terminated when the orbit adjacency graph is pairwise consistent.

To generate a concise, qualitative description of the phase portrait, the orbit adjacency graph is **partitioned** into groups of orbits which are then simplified to produce the **canonical flow patterns**. Each canonical flow pattern consists of a primary orbit and secondary orbits surrounding it. Canonical flow patterns are **classified** into equivalence classes based on a notion of **qualitative equivalence**, which is weaker than topological conjugacy.

The consistency maintenance algorithm, and the procedure to generate qualitative descriptions of phase space dynamics are tested with three different area-preserving maps: the Henon Map, Bak Map, and Standard Map. With some minor extensions to the basic algorithm, KAM is able to automatically reproduce phase portraits and their qualitative descriptions that are similar to those appearing in published papers in the experimental literature.

# Chapter 5

## Parameter Space Searching

To study the dynamics of area-preserving maps, we have been looking at their long-term behavior. That is, we want to see the result of iterating some mapping over and over again, and watch what happens to the orbit starting from some initial state. We have been asking a variety of questions. Given any initial state  $x_0$ , can we predict the nature of orbit starting from  $x_0$ ? For example, does it sit still? Does it repeat itself after some period? Or, does it hop around in the phase space in a random manner? Furthermore, we would like to know what happens if we start the system in a point near  $x_0$ . Does the new orbit stay close to the orbit starting from  $x_0$ ? Does it behave in a similar way? Finally, can we predict what most  $x_0$ 's do? That is, if we pick some starting point randomly, what is the likely result?

In answering these questions, we develop insights about the dynamics of the mapping. This way of thinking about dynamical systems we owe to Poincaré. But there is an extra layer of structure that we have not explored: we wish to see how the pattern of behavior changes as we change the mapping – the equations of motion – a little. It is as though the mapping describes some circuitry inside a black box with a knob that we can turn to adjust some parameter. What we would like to do is to investigate the effects of setting the knob to various values.

Studying the changes in patterns of behavior as some system parameter is varied is the essence of the **Parameter Space Search Problem**. The problem has three parts:

- How are parameter values chosen?
- When is searching in the parameter space terminated?

- How are the search results interpreted and summarized?

The intuitive idea behind the solution is similar to the local consistency test developed in last chapter. Here the interesting object is a phase portrait, a structured collection of orbits. If two adjacent phase portraits with parameter values  $p_1$  and  $p_2$  are *consistent* (and we will say what “consistent” means), there is no need to search inside the interval  $(p_1, p_2)$ . Otherwise, a simple bisection search of the interval is used to look for missing phase portraits that will restore the compatibility between adjacent phase portraits. The consistent phase portraits are then grouped and classified into typical bifurcation patterns according to the bifurcation theory for area-preserving maps.

In general, parameter space searching is a much harder problem than phase space searching. It is more difficult for two reasons. First, it is not clear what is meant by “adjacency of phase portraits” in a multi-dimensional parameter space. Second, we don’t have a complete mathematical theory that classifies all the possible ways in which a phase portrait can change under small perturbations.

The content of this chapter is as follows. The first section introduces the notion of bifurcation. Section 5.2 discusses the Fundamental theorem of Bifurcation Theory. Section 5.3 lists the complete list of generic bifurcations of periodic points of area-preserving maps. Section 5.4 explains how phase portrait compatibility conditions can be derived from knowledge of generic bifurcations. Section 5.5 covers the implementation of the basic consistency maintenance algorithm. The next two sections, 5.6 and 5.7, present experimental results with the Henon map and the Bak map. The chapter concludes with an evaluation of the experimental results, and an end-of-chapter summary.

## 5.1 WHAT ARE BIFURCATIONS?

Phenomena in nature often do one of the four things: they sit still, they grow, they oscillate, or they hop around randomly. A **bifurcation** of a dynamical system is any change in the character of its steady-state behavior as some or all its parameters are changed. Given a small initial push, a mathematical pendulum moving in a frictionless medium will oscillate forever. If damping is “switched on”, the oscillatory behavior will die out and eventually reach a stable rest state (though it may take an infinitely long time to do so). The character of the steady state behavior has changed.

This section explores the notion of bifurcation. But first we should have an intuitive understanding of what parameters are.

Two types of variables – parameters and state variables – appear in the equations describing a dynamical system. The categorization depends on the idea that changes of the system can occur in two separate time scales. Parameters are those variables that change relatively slowly, and to a first approximation are often regarded as constants. State variables, on the other hand, change much more rapidly when compared to the changes of parameters. The familiar mathematical model of the solar system is a good example. Here the state variables are the positions and momenta of the planets relative to the Sun. The masses of the planets and the Sun will be parameters in our model because they change in a much slower time scale than the positions and momenta.

When the parameters of a system are changed gradually, the behavior of the system usually changes in a gradual way, except at certain places where it “jumps”. Think of heating a beaker of water steadily. The temperature rises steadily until it hits 100°C and the water boils. A sudden change in state of matter occurs – water turns into steam. Physicists call this a **phase transition**, a macroscopic change in state due to reorganization at the molecular level. So a gradually changing cause *can* produce a sudden qualitative change in behavior.

As another example, consider adding weights to the top of a vertical, elastic column. With a light weight, the column compresses a little. When the weight is gradually increased, the column suddenly buckles to one side or the other. The vertical equilibrium position of the column has become unstable – a slight disturbance will move the column into a new stable state, one of the two buckled positions. So increasing the loading causes both the number of steady state behavior and their stability properties to change. In other words, a bifurcation has occurred.

Sometimes a bifurcation can move a system from a static equilibrium state to a steady oscillation. Perhaps the most well-known example of such bifurcation occurs in a mathematical model of the vacuum tube which was studied in the 1920s by van der Pol. The vacuum tube is modeled as a simple RLC-circuit except that the resistor is described by a nonlinear relation between the voltage and current. An adjustable parameter, a characteristic of the vacuum tube, controls the resistance or “damping” of the resistor. When the parameter is small, the damping is positive. The system dissipates energy, and one expects the amplitude of current to decay. This means if we start the system near an equilibrium state, it will move towards the equilibrium state. That is, the equilibrium state is stable.

However, as the parameter is increased to some critical point, the damping becomes negative for small amplitude of current. Far away from the equilibrium state, the effect of large amplitude of current still dominates; damping remains positive. It follows that the equilibrium state is not stable because nearby initial states drift away from it. In place of the unstable equilibrium state, an oscillation will build up at certain amplitude where the two effects of opposite tendency – negative and positive damping – balance out. In other words, the system begins to wobble. The basic mechanism for the creation of a steady oscillation when the rest state loses its stability is called a **Hopf bifurcation** [9], after Eberhard Hopf.

Now that we have seen some fairly standard examples of bifurcation, we should seek a deeper understanding of such processes. We'll start with the Henon area-preserving map. Unlike the previous examples, it is not a model of any physical system. It is a purely mathematical construct intended to illustrate the typical bifurcations that a Hamiltonian system with two degrees of freedom can run into. Like any good example, the Henon Map captures the essence of the process in its simplest form, avoiding needless complications.

As noted in Chapter 3, a phase portrait represents the motion of all possible initial conditions corresponding to a given value of the parameter. The phase portrait is made up by fixed points, periodic points, island chains, KAM curves, and chaotic orbits. Now imagine making adjustments to the parameter gradually. The phase portrait changes a little – some island chains may grow while some may shrink, or some KAM curve may become distorted – but qualitatively the main features remain the same. Then, at some critical value of the parameter, something drastic happens to the phase portrait. We may see a fixed point losing its stability, or an island chain disappearing, or a whole new island chain appearing, or a KAM curve breaking up. A bifurcation is taking place. The new phase portrait persists for some range of the parameter values, and another bifurcation occurs. This alternating persistence and bifurcation of phase portraits can repeat infinitely many times.

Let us look at the specific bifurcations that actually occur in the Henon Map. In Fig. 5.1a, we notice a pair of period-5 orbits. One of them is stable – the periodic orbit sits inside a large 5-island chain. The other one is unstable, sitting between successive islands. If we carefully iterate the mapping for different values of  $\alpha$ , we will find that the 5-island chain is created at  $\alpha = 1.27$ , reaches its maximum size at 1.33, and disappears at 1.48.

If we continue increasing  $\alpha$  to about 1.58, two period-4 orbits appear (Fig. 5.1b). Again the stable period-4 orbit is inside a 4-island chain. The island chain grows until

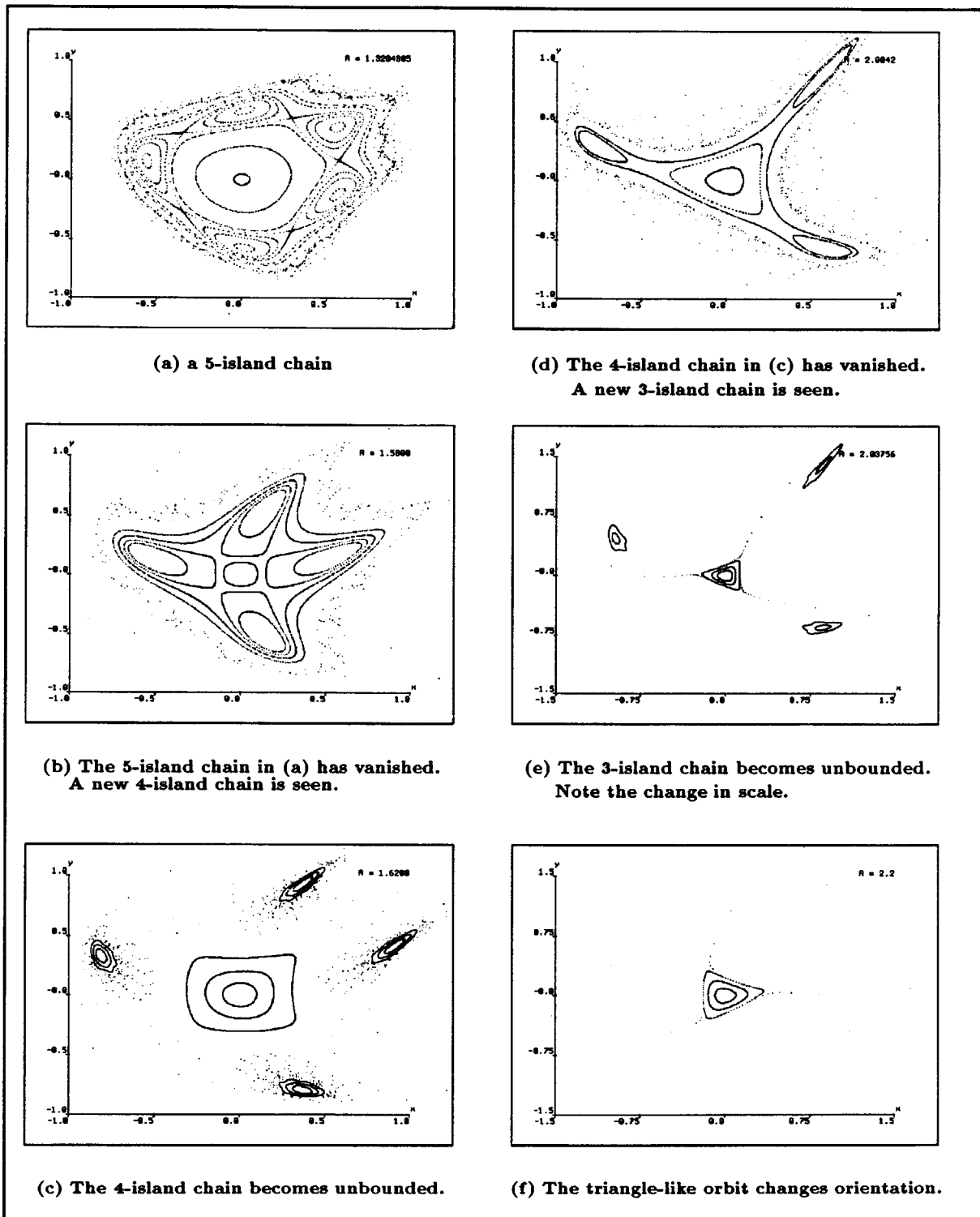


Figure 5.1: *Bifurcations in Henon Map: how the structure of island chains and stability of fixed point change as the parameter  $\alpha$  (or "A" in the figure) is varied.*

$\alpha$  reaches 1.59. Then, the last KAM curve surrounding it breaks up at 1.60. As  $\alpha$  is further increased, the shrinking islands move deeper into the chaotic region. By 1.65 the island chain has vanished.

The range of  $\alpha$  values from 0 to 2 is not very interesting from the point of view of dynamics. Pick any  $\alpha$  in this range, and iterate the mapping. You'll find that the fixed point at the origin remains stable – it is surrounded by closed curves. The island structures around the fixed point may change, but the position and stability of the fixed point do not.

When we increase  $\alpha$  to slightly beyond 2, something dramatic happens. Just inside the tips of the outermost starfish-like orbit, two period-3 orbits appear (Fig. 5.1d). One of these is stable, and gives rise to the 3-island chain. The other one is unstable. But unlike the unstable periodic orbits we have just seen, the three unstable periodic points are placed between each island and the fixed point at the origin. As we keep pushing  $\alpha$ , these unstable points approach the fixed point, collide with it, and emerge again as a new period-3 orbit whose orientation differs from that of the old one by  $180^\circ$  (Fig. 5.1e,f). The fixed point at origin has momentarily turned unstable during the collision. That means, even if we start the system exactly at the fixed point, a tiny push to the system will cause it to move away from the fixed point permanently.

Increasing  $\alpha$  beyond 2.12 does not produce any *new* type of bifurcation. What happens is the regular region around the fixed point at origin continues to grow. Bifurcations of periodic orbits (with period 7 and 12), not unlike that of the period-4 and period-5 orbits that we have seen, occur at approximately 2.8.

A pair of period- $n$  orbits (with  $n \geq 4$ ) appearing near the vertices of a regular  $n$ -gon around the elliptic fixed point, a pair of period-3 orbits bifurcating from the a KAM curve with cusp-like tips, and an unstable period-3 orbit colliding with the elliptic fixed point – these three types of bifurcation are all one can find in our particular map, the Henon Map.

But the Henon Map is special: the fixed point at origin is always stable except at one value of  $\alpha$ . This special property prevents bifurcations due to the loss of stability of the fixed point – e.g., the creation of a stable periodic steady state – to occur. It is therefore instructive to look at a mapping which has a fixed point that loses its stability at some critical parameter value and that remains unstable for a range of parameter values. For this purpose, let us turn to the Bak Map [4] (Fig. refbak-eqn).



**BAK MAP**

$$x_{n+1} = -y_n + x_n(ax_n^2 - a + 2)$$

$$y_{n+1} = x_n$$

Range of  $x$ :  $(-1 \ 1)$   
 Range of  $y$ :  $(-1 \ 1)$   
 Range of  $a$ :  $(0 \ 6)$   
 Escape range:  $(-2 \ 2)$

Figure 5.2: *The Bak Map: a mathematical model of metal-insulator transitions in Peierls systems.*

The Bak Map has a fixed point at the origin. By a simple calculation, we can show that the fixed point is stable for  $a < 4$  and becomes unstable for  $a \geq 4$ . Better still we can see it geometrically by looking at the phase portraits for  $a$  around 4 (Fig. 5.3). When  $a = 3.98$  the fixed point is surrounded by elliptical closed orbits; so the fixed point is stable. However, when  $a = 4.02$  two orbits appear to cross at the origin. This is known as a **saddle** or a **hyperbolic** fixed point. Actually the orbits do not cross; something much more interesting happens (see section 3.1).

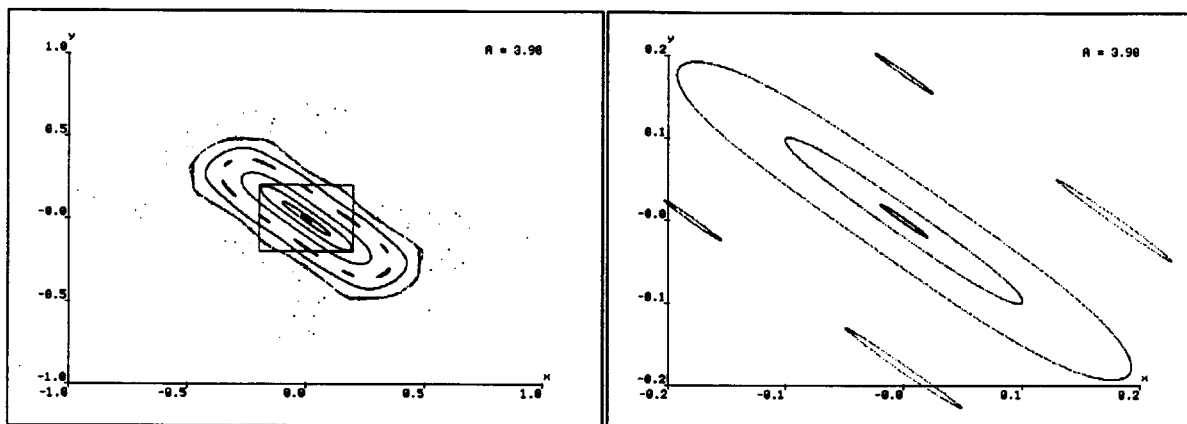
For now let us figure out where the stable steady state goes. If we look at the phase portrait again, we see that there is a period-2 orbit near the unstable origin. Pick a point close to one of the two periodic points. The iterates of that point will flip alternatively between these two periodic points.

If we increase  $a$  to 6 the period-2 orbit also becomes unstable, and a pair of period-*two* orbits appear. One of these period-2 orbits is stable. It too goes unstable at about 6.242, and a period-4 orbit appears (Fig. 5.4). By 6.271 the period has doubled again to eight. By 6.274 it has reached 16, and thereafter we get a rapid sequence of doublings of period to 32, 64, ... One can follow this period-doubling cascade for no more than a few stages because very soon the roundoff error will ruin the accuracy of the computation. <sup>1</sup>

What happens after this infinite sequence of period-doublings is a little bit unclear because as the period doubles, the regular region around the stable periodic orbit is

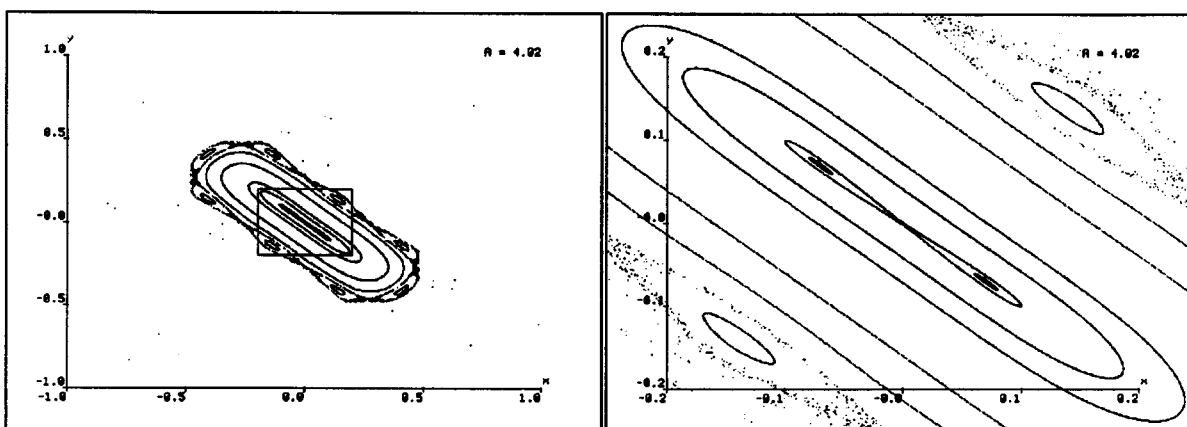
---

<sup>1</sup>If you really insist, I can tell you that period-128 orbit goes unstable at 6.274515062, and the period-256 orbit at 6.2745150609.



(a) Magnify orbits around the fixed point.

(b) The fixed point is about to split.



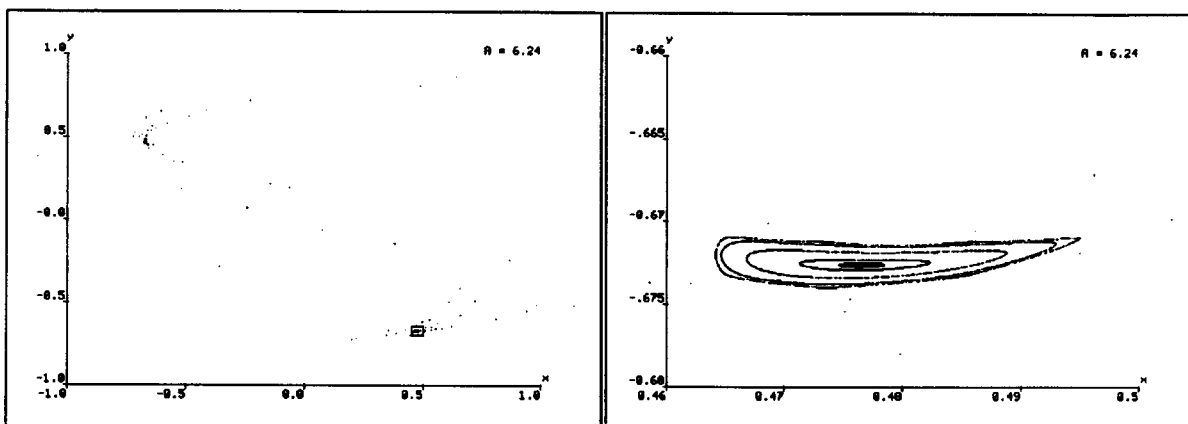
(c) Magnify orbits around the fixed point.

(d) The fixed point has become unstable. A new period-2 orbit is created.

Figure 5.3: *Period doubling in the Bak Map: bifurcation of a fixed point into a period-2 orbit.*

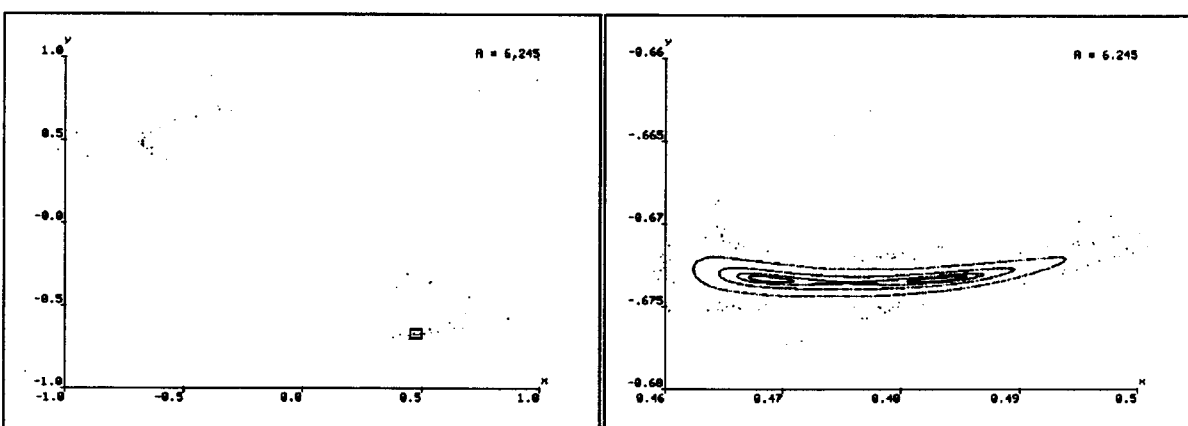
shrinking rapidly. When the period has doubled infinitely often, the fraction of phase space occupied by island chains is so tiny that following them numerically becomes problematic.

In a few pages we will see that, roughly speaking, one-parameter area-preserving maps possess some or all of these bifurcations, and typically nothing else. Several of these bifurcations can coexist, but we won't find anything more complicated. We will discuss these four types of bifurcation; but first, we need an important theorem in bifurcation theory which tells us *when things don't change*.



(a) Magnify orbits around one of the period-2 points

(b) The period-2 point is about to split.



(c) Magnify orbits around the bottom part of the period-4 orbit

(d) The period-2 orbit has become unstable. A new period-4 orbit is created.

Figure 5.4: *Period doubling in Bak Map: bifurcation of a period-2 orbit into a period-4 orbit.*

## 5.2 CORE IDEA: STRUCTURAL STABILITY

A bifurcation is any change in the qualitative properties of a dynamical system. Bifurcation theory is concerned with the change in the number of steady state solutions, their stability, and how they are connected as the governing equations are changed. It is an active research area in Dynamical Systems Theory because many questions are still open. For instance, we have little idea about how complicated steady state solutions – e.g., chaotic orbits, bifurcate. We also know very little about global bifurcation,

which studies how the connections among steady state solutions in the phase space can change. So in this section I will confine the discussion to the simplest cases of bifurcation: bifurcation of fixed points and periodic orbits.

Now one may ask a very natural question: what is the list of typical ways for a fixed point of an area-preserving map to bifurcate? By examining the Henon map and Bak Map, we had already made a start on this. We have found four possible bifurcations: a pair of period- $n$  ( $n \geq 4$ ) orbits surrounding the fixed point, a pair of period-3 orbits bifurcating from a closed curve, the fixed point colliding with an unstable period-3 orbit, and period-doubling.

But how do we know these four are all the *typical* bifurcations? It's not hard to imagine other types of bifurcation. For instance, might not a fixed point lose its stability to a pair of stable fixed points instead of a stable period-2 orbit? Or, similarly why aren't period-tripling or period-quadrupling (or other period-multiplying) typical?

To resolve questions like these, one must have a clear sense of the word "typical". The idea is that it is not true that any bifurcations that can occur in area-preserving maps must be one of these four. For example, it is easy to concoct systems with special symmetries that have more complicated bifurcations. The Bak Map, which has Rimmer bifurcations, is a good example. If one tries to list all *possible* bifurcations that can happen, one will soon find that the task is infinitely difficult. There will be infinitely many different kinds of bifurcations, and most of them will be very complicated. It is far simpler to analyze those that arise often, and avoid the rare exceptional ones.

Another caveat: what is typical depends on the space of systems you are talking about. Bifurcations typical in an area-preserving map behave quite different from those typical in a non-area-preserving map. And bifurcations typical in a many-parameter area-preserving map, may not be typical in a one-parameter area-preserving map.

Enough precautions. Let us turn to the typical bifurcations of area-preserving maps. This discussion could be made perfectly precise and much more elegant – but at the expense of technicalities, like normal forms, that are not suitable for this level of discussion. I'll explain all the typical bifurcations of a fixed point, and how one can tell if a given map is undergoing one of these bifurcations.

The idea is to study some simple systems that we completely understand and use them to understand complicated systems that we don't. The simplest area-preserving maps are the *linear* ones. From linear algebra, we know that linear mappings can be

represented as matrices. Moreover, the theory of Jordan Canonical Forms tells us that any real 2 x 2 matrix is related by a similarity transformation to a simple matrix that has one of the three canonical forms:

$$\begin{array}{ccc} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} & \begin{pmatrix} a & -b \\ b & a \end{pmatrix} & \begin{pmatrix} \lambda & 0 \\ 1 & \lambda \end{pmatrix} \\ I & II & III \end{array}$$

Linear area-preserving maps form a subclass of all linear transformations. If such a mapping is used to transform any region of the phase space, its area remain unchanged after the transformation. That means, the determinant of the transformation must be 1. This special property implies a constraint on the eigenvalues, namely, the product of the eigenvalues must also be 1. So the eigenvalues are either real numbers ( $\lambda$  and  $\frac{1}{\lambda}$ ) or complex conjugates on the unit circle of the complex plane.

To get a feel of what these transformations do, we let them act on a unit square in the euclidean plane (Fig. 5.5).

For type I, real and distinct eigenvalues, let's assume  $\lambda_1 > 1 > \lambda_2$ .<sup>2</sup> (The case when the  $\lambda$ s are negative is explained in a similar way.) Under the action, the unit square is stretched horizontally, and squeezed vertically. The eigenvalues control the amount of stretching or squeezing. Mathematicians have generalized this idea of stretching and squeezing to higher-dimensional systems. Mappings that stretch any region in phase space along some directions, and squeeze along the remaining directions are called **hyperbolic**. The notion of hyperbolicity is quite general; it applies to not just Hamiltonian systems, but dissipative systems as well.

The hyperbolic maps have a very important property: they are **structurally stable**. The idea was invented by Aleksandr Andronov, a Russian mathematician, in the 1930s. A structurally stable system is one whose behavior does *not* change if you make sufficiently small change to the equations. That means, any system close to the original system will have the same number of fixed points and periodic points. Moreover, the stability of the fixed points and periodic orbits won't change.

We need to be careful here. Structural stability is a quite different idea from stability of a steady state of a system. A steady state is stable if it is stable with respect to

---

<sup>2</sup>Recall that the eigenvalues of a diagonal matrix are the diagonal elements.

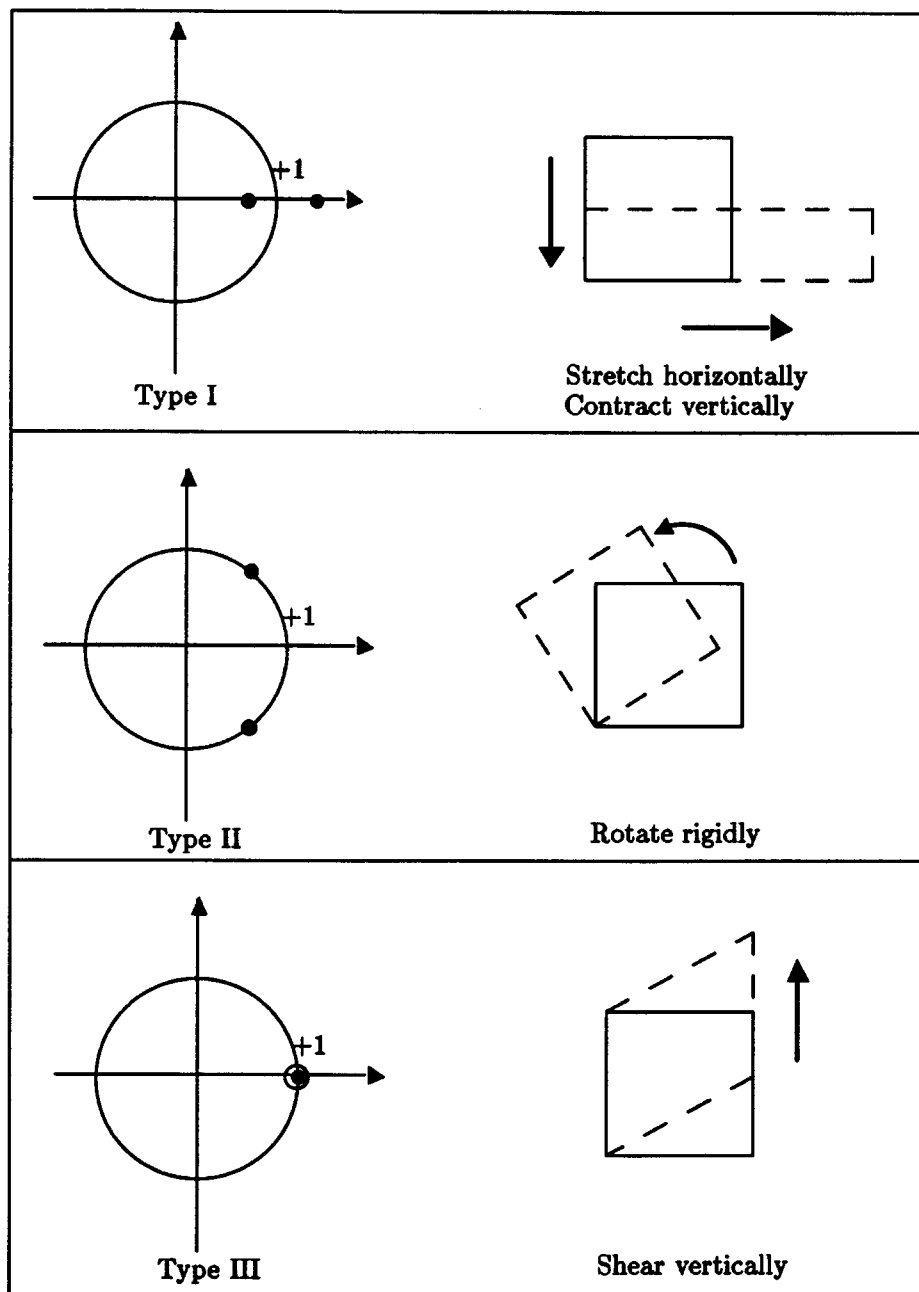


Figure 5.5: *How linear area-preserving maps act on a unit square*

small changes in the initial condition. But structural stability is a property of the whole system. A system is structurally stable if it is stable with respect to small changes in

the equations. So, for instance, the origin is an unstable fixed point of a type I map, but the map itself is structurally stable. A small perturbation to the map may cause the eigenvalues to move around a bit, but the stretching and squeezing actions still remain.

For type II, complex eigenvalues with unit modulus, the action is a rigid rotation. The square is rotated by some angle about the origin. Are area-preserving maps with complex eigenvalues structurally stable? This is a natural question to ask. But the answer is no.

To see why, let us consider the case when the angle of rotation is some rational multiple of  $2\pi$ . Here we take the unit of angular rotation to be  $2\pi$  radian. So let's say the angle of rotation is  $\frac{1}{3}$ . Then after 3 successive rotations the unit square will come back exactly to where it started. In general, if the rotation is  $\frac{m}{n}$ , then every point will map to itself after  $n$  applications of the transformation. In other words, we have lots of period- $n$  orbits around.

Suppose the map is perturbed a little. It is still acting like a rigid rotation because the eigenvalues are still complex. But the angle of rotation need not be rational anymore. It can become irrational. That means, instead of periodically coming back to the same position, the motion will never repeat exactly. It does however almost repeat in the sense that a point can come arbitrarily close to the initial state. This is why mathematicians use the word "quasiperiodic" to describe this type of motion.

Now we can see why a type II map is not structurally stable. A tiny change to the equations may cause a system that rotates with a rational angle to lose all its periodic orbits, and moves in a quasiperiodic manner.

For Type III, real and equal eigenvalues with modulus 1, the action is a shearing. The unit square is sheared in the  $y$ -direction. Mappings having a double eigenvalue are quite atypical: tiny changes to the equations will separate the two eigenvalues, creating either a type I or a type II map. If type I and type II are coins landing heads or tails, then type III is a coin landing on its edge. In principle, it can happen. In practice, it doesn't.

What do we learn from these simple mappings? That eigenvalues of a linear mapping are the important quantities to look at. First, they determine what type of action the mapping induces. Second, mappings with eigenvalues not on the unit circle of the complex plane are the structurally stable ones.

These two observations are remarkable because it turns out that they hold true even for nonlinear mappings under some reasonable conditions. I'll be more precise about

this. But first we have to generalize the idea of eigenvalue to nonlinear mappings. We know what eigenvalues of linear mappings are. To deal with a nonlinear mapping, one somehow approximates it – at least locally – by a linear mapping. A familiar trick in mathematics. Since the behavior around fixed points are the simplest to understand, it is natural to study the relationship between the linearization of the mapping at a fixed point and the full mapping itself. The hope is that the eigenvalues of the linearization of the mapping about a fixed point – the **multipliers** of the fixed point, for short – will tell us a lot about the behavior of the actual mapping, at least in a sufficiently small region around the fixed point.

We can't discuss the relevant theorem in this section until we have a clear idea of what is meant by “changing a map a little”. The standard mathematical setup is to use a family of mappings depending smoothly on a (real) parameter  $\alpha$ .

Suppose  $g_\alpha : R^n \rightarrow R^n$  is a smooth one-parameter family of mappings, and  $x_0$  is a fixed point of  $g_\alpha$ . In other words, at one parameter value  $\alpha_0$ , the map has a fixed point at  $x_0$ . The questions we'd like to ask in Bifurcation Theory are: What does a map  $g_\alpha$  look like for some  $\alpha$  close to  $\alpha_0$ ? Does the new map have a fixed point? If so, is the fixed point like  $x_0$ ? Is it near  $x_0$ ?

In the early 1960s, Stephen Smale and Philip Hartman proved a general theorem [9] – it applies to both Hamiltonian and non-Hamiltonian systems – which implies, among other things, that if  $g_{\alpha_0}$  has a fixed point  $x_0$  with no multipliers equal to  $+1$ , then any  $g_\alpha$  with  $\alpha$  close to  $\alpha_0$  will have the same number of fixed points near  $x_0$ . This is a powerful theorem. It says if we compute the multipliers of a fixed point and find out that none of them is  $+1$ , then we can conclude that a tiny change to the mapping will leave the number of fixed point unchanged. In other words, the original fixed point will not vanish, and no new one appears as we perturb the mapping a little.

The theorem, however, does not guarantee that the stability of the fixed point will be unchanged. We have already seen an example of this in the Henon Map. Around  $\alpha = 2$ , the multipliers of the fixed point at origin are purely imaginary:  $+0.333i$  and  $-0.333i$ . They satisfy the hypothesis of the theorem. We therefore expect that for  $\alpha$  close to 2, Henon map should have one fixed point near origin. This is true. In fact, the fixed point is always at the origin. On the other hand, we have seen that the fixed point momentarily turns unstable when it collides with a period-3 orbit. So the stability of the fixed point is not preserved.

It turns out that a simple extension to the theorem guarantees that both the number and stability of the fixed point will remain unchanged under small perturbation. The



stronger version goes like this. If the  $g_{\alpha_0}$  has a hyperbolic fixed point  $x_0$ , i.e., none of its multipliers has modulus 1, then any  $g_\alpha$  with  $\alpha$  close to  $\alpha_0$  will have the same topological structure locally around the fixed point. In other words, hyperbolic fixed points don't change under small perturbation; they are structurally stable properties.

The Smale-Hartman theorem is so important that some dynamical theorists refer to it as the **Fundamental Theorem of Bifurcation Theory**. Essentially it tells us when nothing interesting happens. So to look for bifurcations we'd better focus on non-hyperbolic fixed points.

### 5.3 GENERIC BIFURCATIONS OF FIXED POINTS

“Generic” is just a fancy term for “typical”. Mathematicians have found the idea of ignoring the rare, degenerate cases – situations that can arise in principle, but not in practice – so useful that they feel the need for inventing a technical term for it. In this section, I will go through the list of generic bifurcations of fixed points. This piece of mathematical knowledge is essential for KAM because it is the basis for KAM's automatic search and interpretation for the bifurcation patterns.

The Fundamental Theorem is a valuable tool in Bifurcation Theory; it has some interesting consequences.

The first major consequence is that a bifurcation theory for fixed points of mappings will give us, for free, a bifurcation theory for periodic points of mappings. The reason is simple: a period- $n$  point of a mapping is the same as the fixed point of a new mapping that is obtained by composing the original mapping with itself  $n$  times. In other words, suppose  $x_0$  is a period- $n$  point for a mapping  $g_{\alpha_0}$ , i.e.,  $g_{\alpha_0}^n(x_0) = x_0$ . Define a new mapping  $f_{\alpha_0} = g_{\alpha_0}^n$ . Then we have  $f_{\alpha_0}(x_0) = x_0$ .

So if we want to study the status of the period points of  $g_{\alpha_0}$ , we can equivalently study the status of the fixed point of  $f_{\alpha_0}$ . Using the Fundamental Theorem, we conclude that the persistence and stability of the periodic points of  $g_{\alpha_0}$  are determined by the multipliers of the fixed point of the mapping  $g_{\alpha_0}^n$ . In particular, if none of the multipliers of  $x_0$  is  $+1$ , then the number of the periodic orbits for  $g_\alpha$  is unchanged under small perturbation.

This relationship between periodic orbits and fixed points also gives us a hint regarding the classification of typical bifurcations. The idea is that whenever any multiplier of

a fixed point of  $g_{\alpha_0}^n$  is equal to  $+1$ , the number of period- $n$  orbits can change. In other words, new periodic orbits can appear whenever any multiplier of a *periodic point* of  $g_{\alpha_0}$  passes through an  $n$ th root of unity. Recall the first root of unity is  $+1$ , the second roots are  $+1$  and  $-1$ , the third are  $+1$ ,  $e^{2\pi i \frac{1}{3}}$ ,  $e^{2\pi i \frac{2}{3}}$ , and so on.

Let us acquaint ourselves with the list of generic bifurcations now.

### $\lambda = +1$ : Extremal Bifurcation

A fixed point can lose its stability by colliding with an unstable fixed point. This is known as the **extremal bifurcation** (Fig. 5.6). At the moment of collision both fixed points disappear, and a quasiperiodic orbit with cusp-like tips is formed. The number of fixed points changes after the bifurcation.

Let us see what happens. As the parameter is varied, the stable fixed point gets closer to the unstable one. Its domain of stability, i.e., the set of points whose orbits will stay close to the fixed point, shrinks. That means, a small perturbation to the system will pull it away from the neighborhood of the fixed point. The system will go to some new stable steady state which need not remain close to the old equilibrium position. Physicists describe this situation as a *hard* loss of stability because a smooth change of the parameter results in a stiff “jump” of the motion.

If the parameter is changed in another direction, then the same phenomenon is described by the birth of a pair of fixed points – one stable, the other unstable – from a quasiperiodic orbit with cusp-like tips.

### $\lambda = -1$ : Period-Doubling Bifurcation

**Period-doubling** (Fig. 5.6) describes a second way in which a stable fixed point can lose its stability: it does not “die” but instead transfers its stability to some periodic steady state. As the parameter is varied, the fixed point is becoming hyperbolic. At the critical parameter value where the fixed point turns hyperbolic, a stable period-2 orbit is created. The system moves from a stable equilibrium to a stable periodic steady state. Unlike the extremal bifurcation, the loss of stability here is *soft* because immediately after the bifurcation the oscillating behavior stays close to the fixed point. In this sense, the behavior of motion changes continuously.

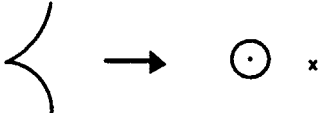
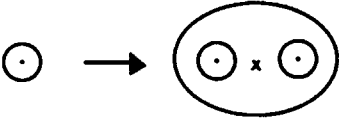
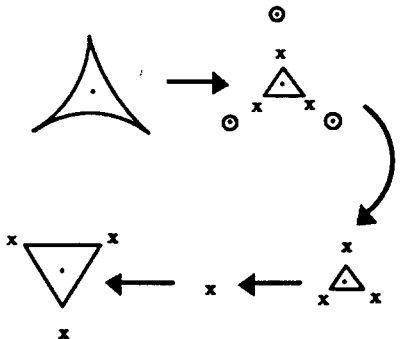
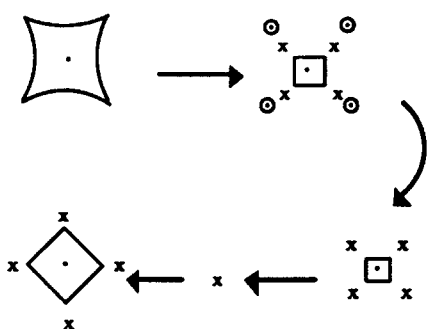
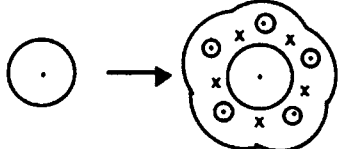
Bifurcation Type	Flow Pattern
extremal bifurcation	
period doubling	
phantom 3-kiss	
phantom 4-kiss	
Poincaré-Birkhoff	

Figure 5.6: Five generic bifurcations for one-parameter area-preserving maps. Dots ( $\bullet$ ) indicate elliptic fixed points. Circles with a dot inside ( $\odot$ ) stand for islands. Crosses ( $\times$ ) are hyperbolic fixed points.

Notice that unlike the extremal bifurcation the number of the fixed point does not change during the period-doubling. This follows from the Fundamental Theorem because the multiplier is not  $+1$ .

$\lambda = e^{2\pi i \frac{1}{3}}$  : **Phantom 3-Kiss**

There is a third way for a fixed point to lose its stability, namely, by colliding with an unstable period-3 orbit. This type of bifurcation is called a **phantom 3-kiss** (Fig. 5.6). “3-kiss” seems clear enough: three periodic points kissing the fixed point. The word “phantom” seems to convey some devilish consequence of the kissing – the fixed point is no longer visible as its domain of stability vanishes. Well that’s my theory anyway.

The phantom 3-kiss also results in a hard loss of stability. The system jumps to a new steady state that can be arbitrarily far away.

$\lambda = e^{2\pi i \frac{1}{n}}, n \geq 5$ : **Poincaré-Birkhoff Bifurcation**

Poincaré-Birkhoff bifurcation occurs most frequently. It happens when the multiplier of a fixed point goes through an  $n$ th root of unit ( $n \geq 5$ ). The result is the creation of a pair of period- $n$  orbits: one stable, one unstable. The stable periodic points are embedded inside an  $n$ -island chain (Fig. 5.6) surrounding the fixed point. The unstable periodic points sit roughly in between two successive “tips” of the islands. To understand this, let us turn to the mechanism behind the creation of the island chain.

If the mapping is linear, then every point in the phase space will repeat itself after  $n$  applications of the mapping. So there are infinitely many period- $n$  orbits, and they lie on concentric circles about the fixed point. Suppose you add a small amount of nonlinearity to the mapping in such a way that the new mapping is still area-preserving. What will happen to these circles of periodic points?

To see what happen, let’s split the nonlinearity into two components: tangential and radial. This is useful because these two components have different effects. The tangential component affects the amount of rotation of the circles; the radial distorts the shape of them.

Consider first the effect of the tangential perturbation. It changes the amount of rotation on the circles depending on their radial distances from the center. For instance,

circles farther away may rotate with a larger angle than those near the center. Let's call the amount of rotation the **rotation number** of the circle. In a small region around the fixed point, we will assume the rotation numbers change continuously as a function of the radial distance. This is not a very restricted assumption because a large class of mappings, called **twist mappings**, have this kind of behavior; they map circles into circles but with the rotation number (i.e., the twisting) dependent on the radius of the circle.

With tangential perturbation alone, we will have one instead of infinitely many circles that have rotation number exactly equal to  $\frac{1}{n}$ . But there are still infinitely many period- $n$  points left because every point on this particular circle is periodic. Now consider adding the radial perturbation, which alters the shape of the circles. We might expect these periodic points all to be destroyed. It turns out this is not the case. In general, at least one pair of period- $n$  orbits survives the perturbation. This statement is in essence the content of a theorem first conjectured by Poincaré and later proved by George Birkhoff in the 1930s [14].

$\lambda = e^{2\pi i \frac{1}{4}}$  : **Phantom 4-Kiss/Poincaré-Birkhoff**

What makes this situation interesting is that there are *two* types of bifurcation associated with a multiplier passing through a fourth root of unity. Which type will occur depends on the higher order nonlinear terms of the mapping.

The first type is called the Phantom-4-kiss (Fig. 5.6). It is not really a new type of bifurcation. It looks just like the Phantom-3-kiss except that an unstable period-4 orbit, instead of a period-3, approaches and collides with the fixed point. Like the "3-kiss", the "4-kiss" also results in a hard loss of stability.

The second type is a Poincaré-Birkhoff bifurcation of a 4-island chain. Unlike the Phantom-4-kiss, the fixed point remains stable during the bifurcation.

**Typically, that's it**

Kenneth Meyer proved a theorem [19] to the effect that typically only these five types of bifurcation occur in a one-parameter area-preserving map. Physicists often call the first four types – extremal, period-doubling, phantom 3-kiss, and phantom 4-kiss –

bifurcations under *strong resonances*. “Strong” in the sense that the stability of the fixed point is destroyed by the bifurcation. The Poincaré-Birkhoff bifurcation, on the other hand, corresponds to *weak resonances* because the creation of a pair of periodic orbits does not change the stability of the fixed point that undergoes bifurcation.

## Rimmer Bifurcation

One final type of fixed point bifurcation deserves attention: the Rimmer Bifurcation, after Russell Rimmer who proves that such bifurcation is typical for **reversible mappings** [26], which are mappings that have special symmetry properties (see section 4.7.2). Here at the point of bifurcation, a stable fixed point splits into two new stable fixed points – one on each side of the original fixed point. The picture looks like a period-doubling except that two new fixed points appear instead of one new period-2 orbit. Thus before the bifurcation, the system has one stable fixed point; after the bifurcation, it has three – two stable, one unstable.

The Rimmer Bifurcation also has a weak-resonance analogue: it gives rise to twice as many periodic orbits as the Poincaré-Birkhoff bifurcation. In the Bak Map, for example, it is observed that the fixed point at the origin spits out an  $n$ -island chain when the multiplier passes through an  $n$ th root of unit where  $n$  is odd, but two alternating  $n$ -island chains where  $n$  is even.

Rimmer bifurcations are not *typical* in a general area-preserving map. However, it is often found in mappings with odd symmetry, i.e.,  $f(-x,-y) \rightarrow -f(x,y)$ . An example of odd symmetry is the Bak Map. The property of odd symmetry implies the origin is a fixed point for all parameter values. Since the extremal bifurcation contain parameter values for which there is no fixed point near the point of bifurcation, the type of bifurcation that occurs in such situation ( $\lambda = +1$  with symmetry) is going to be qualitatively different from that without any symmetry conditions imposed.

## 5.4 PHASE PORTRAIT COMPATIBILITY

Bifurcation theory involves many difficult mathematical theorems. To capture all of this knowledge in a computer program is not a simple task. Fortunately, applying the classification results of generic bifurcations does not depend on a detailed understanding

of the proofs of these theorems. In this section, I will describe how the knowledge of the typical bifurcations allows us to formulate simple rules that decide when two flow patterns are inconsistent. We say two flow patterns are **consistent** if the difference in their topological structures can be explained by a typical bifurcation.

### 5.4.1 Pairwise Portrait Consistency

The key strategy that underlies KAM's search for bifurcation patterns in the parameter space is to focus the search in regions where the neighboring phase portraits have flow patterns that are inconsistent. I will explain the rules for deciding flow pattern inconsistency. But first, let me describe a few more things about the input to the program that carries out parameter space searching: a **canonical flow pattern**.

The canonical flow pattern was introduced in Chapter 4 as a data structure for representing the essential qualitative features of a phase portrait. For example, orbits that are redundant – such as the non-maximal island chains, and the non-maximal quasiperiodic orbits – are removed to simplify the description. A canonical flow pattern is a group of essential orbits surrounding some primary orbit. The primary orbit can be a stable fixed point or a stable periodic orbit. In the former case, the canonical flow pattern is denoted by the symbol F-FLOW-PATTERN; in the latter, by P-FLOW-PATTERN. Within each of these two categories, a canonical flow pattern is further characterized by four indices  $(m, n, p, q)$  which, among other things, indicate the number of island chains it has, and the number of those island chains that are bounded by some closed curve.

For an example, consider the flow pattern in Fig. 5.7. 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 F-FLOW-PATTERN-2-1. For brevity, I'll usually drop the prefix "F-" if it is unambiguous in context as to which category of flow pattern I am referring. 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  $(5\ 6)$ . If the flow pattern does not have any island chain, then the period is NIL, the empty list.

We also need a measure of how large a flow pattern is. The **area** of a flow pattern is the area, in some normalized unit, of the region in phase space that is occupied by quasiperiodic orbits or island chains. In this example, the area is obtained by summing the area enclosed by the closed curve, and the area enclosed by the 6-island chain. If

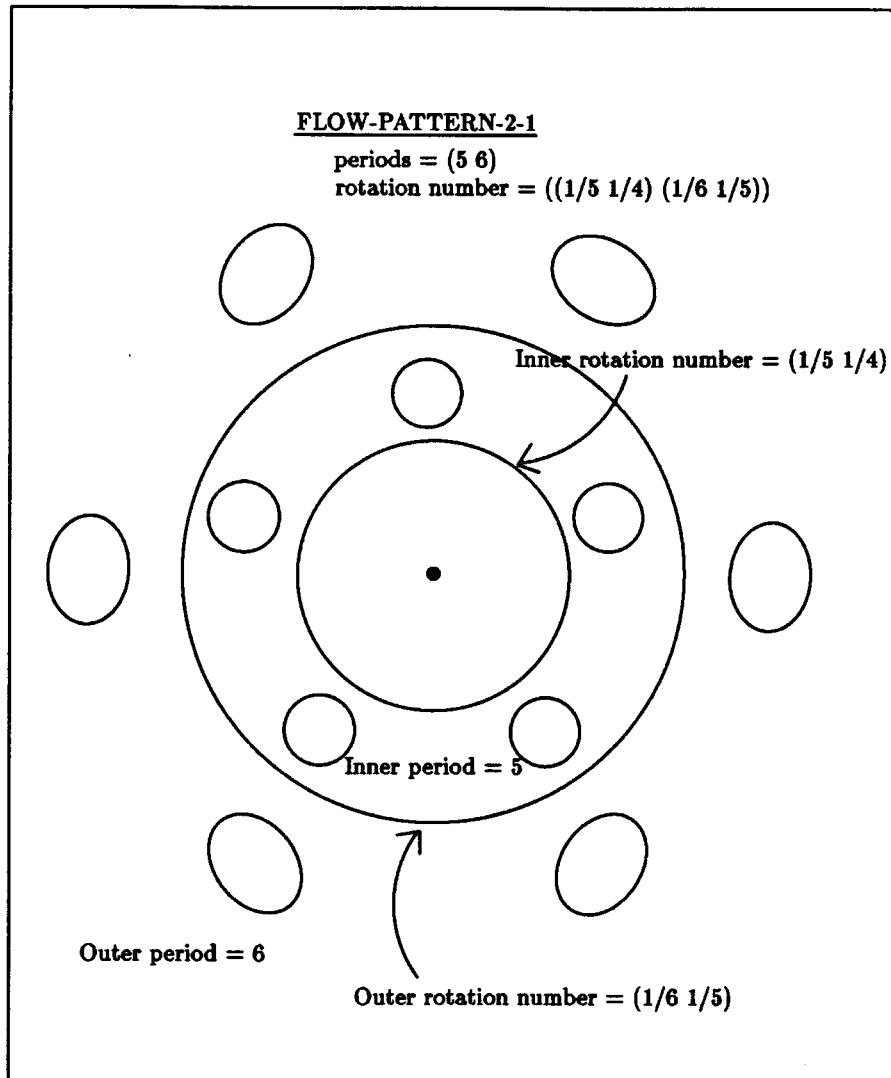


Figure 5.7: This picture illustrates two important quantities of a flow pattern: period and rotation number. The flow pattern has a 5-island chain and a 6-island chain. Its period is (5 6). Its rotation number is the list of rotation number bounds of its maximal quasiperiodic orbits.

the flow pattern does not have any unbounded island chain, then the area is simply that enclosed by the outermost closed curve surrounding the fixed point.

One final quantity is needed: the **rotation number** of a maximal quasiperiodic orbit. Rotation number is an important quantity because it tells us where to look for an island



chain (see section 4.3). The rotation number of a flow pattern is the list of all rotation number bounds for its maximal closed orbits. Take the flow pattern in Fig. 5.7 as an example. It has two maximal closed orbits. The inner one has rotation number  $\in (\frac{1}{5}, \frac{1}{4})$ ; the outer one  $\in (\frac{1}{6}, \frac{1}{5})$ . So the rotation number of the flow pattern is  $((\frac{1}{5}, \frac{1}{4}), (\frac{1}{6}, \frac{1}{5}))$ .

Two rotation numbers are called **farey adjacent** if there is no rational with a small denominator ( $\leq 8$  for this implementation) in the farey ordering. In this example, the two rotation numbers are not farey adjacent because a rational with small denominator,  $\frac{1}{5}$ , is between them. When two quasiperiodic orbits have farey adjacent rotation numbers, there is not going to be any low-order island chain between them.

I will divide the rules for determining inconsistency into two groups: (1) those that are useful heuristics, and (2) those that are consequences of the bifurcation theory.

### Inconsistency Rules that are heuristics

Fig. 5.8 shows the two heuristic rules for determining flow pattern inconsistency. The first one is the **large size difference rule**, which says if two flow patterns differ greatly in size, then they are not consistent.

The second rule introduces the term “maximal flow pattern”. This is just a fancy term for a flow pattern that does not sit between two other flow patterns. In other words, given a sequence of flow patterns ordered by their parameter values, the maximal ones are the first and last one in the sequence. The purpose of the second rule is to extend the search to cover the part of the parameter region near one of its endpoints.

The non-maximal parameter rule is not invoked until all the pairwise inconsistency rules have been tried. This rule ordering gives a higher priority to finding missing flow patterns between inconsistent neighboring flow patterns.

The next group of rules are based on knowledge of rotation number and generic bifurcations. I will divide them into two sub-groups: the strong resonance bifurcations, and the weak ones.

The first sub-group has four rules (Fig. 5.9). The first rule is a general rule. The remaining three are the strong resonance rules. The **qualitative equivalence rule** says that two flow patterns are consistent only if they are qualitatively equivalent (i.e., have the same primary and secondary orbits), and their respective rotation numbers are farey adjacent.

<p><b>RULE 001    Large size difference</b>  If    A and B have very different sizes (more than a factor of 3)  then they are not consistent</p> <p><b>RULE 002    Non-maximal parameter</b>  If    (1) A is a maximal flow pattern        (2) the distance between the parameter value of A and the closest            parameter bound is less than certain threshold (1%)  then A is not consistent</p>
---

Figure 5.8: *Two heuristic rules for determining flow pattern inconsistency.*

The **period-doubling rule** says a F-FLOW-PATTERN is not consistent with a P-FLOW-PATTERN unless the primary orbit of the P-FLOW-PATTERN is periodic with period 2.

For the phantom-3-kiss and the phantom-4-kiss bifurcation, we notice that immediately before and after the critical parameter value at which collision occurs, the flow patterns are qualitatively equivalent, and their rotation numbers are farey adjacent. The only difference is that the orientation of the pattern changes. For example, the triangle-like closed curve in the phantom-3-kiss rotates 180° after the collision. The purpose of the **phantom-3-kiss rule** and the **phantom-4-kiss rule** is to allow KAM to look for the flow pattern corresponding to the critical parameter value.

Computing the orientation of a triangle-like or a square-like closed curve requires two steps. First, find the locations on the curve that have locally maximal curvature. They correspond to the “tips” of the curve. Second, calculate the angular positions of these locations with respect to the horizontal axis. The **orientation** of the closed curve is defined as the mean of these angles. The orientation of a flow pattern is then the orientation of its outermost closed curve.

The phantom-3-kiss rule says if the orientations of the two flow pattern differ by more than 45°, then they are not consistent. The phantom-4-kiss rule has a similar explanation.

The final group of rules (Fig. 5.10) implements the idea behind the Poincaré-Birkhoff bifurcation. It requires that two neighboring flow patterns differ by at most one in the number of island chains that they have. There are a few details in these rules concerning

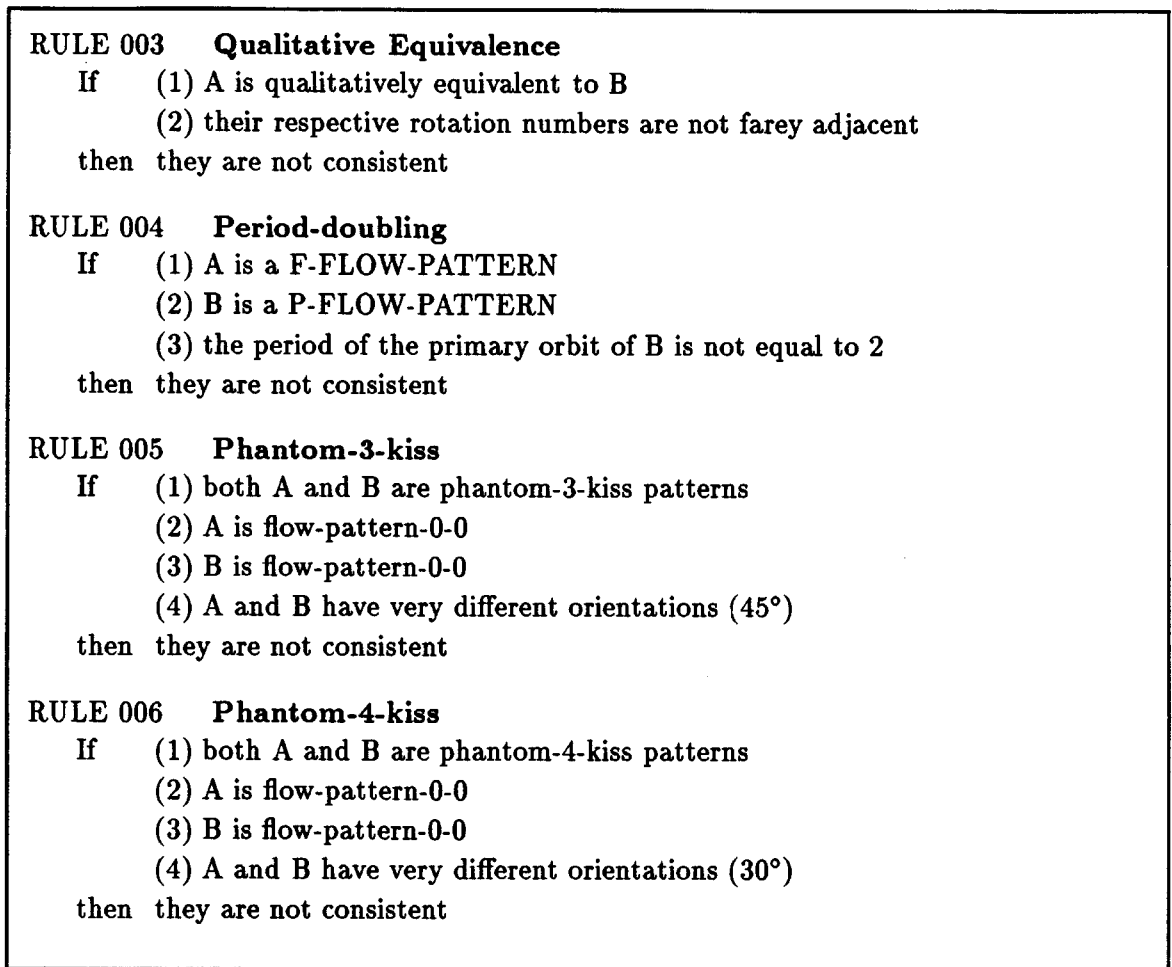


Figure 5.9: *Inconsistency Rules for Qualitative equivalence, and the strong resonances.*

the number of unbounded island chains and rotation numbers, but we don't have to worry about them now.

We have seen all the inconsistency rules for F-FLOW-PATTERN. There is an analogous set of rules for P-FLOW-PATTERN, but they do not involve any new idea.

### 5.4.2 Suggestion Rules

The purpose of the suggestion rules is to propose new candidate parameter value for further exploration. Since the search space is one-dimensional, the suggestion rules here are much simpler than those for phase space searching. Two suggestion rules seem to

<p><b>RULE 007 Poincare-Birkhoff-1</b>  If (1) A is flow-pattern-2-2  (2) B is flow-pattern-2-1  (3) A and B do have the same periods  then they are not consistent</p> <p><b>RULE 008 Poincare-Birkhoff-2</b>  If (1) A is flow-pattern-2-2  (2) B is flow-pattern-1-1  (3) the period of B is not the same as any of the periods of A  then they are not consistent</p> <p><b>RULE 009 Poincare-Birkhoff-3</b>  If (1) A is flow-pattern-2-1  (2) B is flow-pattern-2-0  (3) A and B do have the same periods  then they are not consistent</p> <p><b>RULE 010 Poincare-Birkhoff-4</b>  If (1) A is flow-pattern-2-1  (2) B is flow-pattern-1-1  (3) the period of B is not the same as the inner period of A  then they are not consistent</p> <p><b>RULE 011 Poincare-Birkhoff-5</b>  If (1) A is flow-pattern-2-1  (2) B is flow-pattern-1-0  (3) the period of B is not the same as the outer period of A  then they are not consistent</p> <p><b>RULE 012 Poincare-Birkhoff-6</b>  If (1) A is flow-pattern-2-0  (2) B is flow-pattern-1-0  (3) the period of B is not the same as the inner period of A  then they are not consistent</p>
---

Figure 5.10: *Inconsistency Rules for Poincare-Birkhoff bifurcations.*

<p><b>RULE 013 Poincare-Birkhoff-7</b>          If (1) A is flow-pattern-1-1          (2) B is flow-pattern-1-0          (3) the period of B is not the same as the period of A          then they are not consistent</p> <p><b>RULE 014 Poincare-Birkhoff-8</b>          If (1) A is flow-pattern-1-1          (2) B is flow-pattern-0-0          (3) the outer rotation number of A is not farey adjacent          to the rotation number of B          then they are not consistent</p> <p><b>RULE 015 Poincare-Birkhoff-9</b>          If (1) A is flow-pattern-1-0          (2) B is flow-pattern-0-0          (3) the rotation numbers of A and B are not farey adjacent          then they are not consistent</p> <p><b>RULE 016 Poincare-Birkhoff-10</b>          If (1) A has two island chains          (2) B has no island chains          then they are not consistent</p> <p><b>RULE 017 Poincare-Birkhoff-11</b>          If (1) A has two bounded island chains          (2) B has no bounded island chains          then they are not consistent</p> <p><b>RULE 018 Poincare-Birkhoff-12</b>          If (1) A is flow-pattern-2-0          (2) B is flow-pattern-1-1          then they are not consistent</p>
---

Figure 5.10: (cont.) Inconsistency Rules for Poincare-Birkhoff bifurcations.

suffice. The first rule handles pairwise inconsistent flow patterns. It simply suggests the midpoint of the parameter values of the two offending flow patterns. The second rule handles a maximal flow pattern. Again, it suggests the midpoint between the parameter

value of the offending flow pattern and the closest boundary of the parameter space.

## 5.5 IMPLEMENTATION

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. 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 and singly consistent.

However, there are two major differences between the two problems that make the parameter space search algorithm easier to implement. First, we have restricted the parameter space to be a one-dimensional space. Among other things, this restriction makes the problem of determining adjacency trivial. This is so because in 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 relation can be determined algebraically without resort to geometric algorithms like the contour activation for deciding orbit adjacency. Moreover, searching for missing objects is simple because the one-dimensional bisection search runs in logarithmic time and is easy to implement.

Second, whereas an orbit can be a curve or a region in the phase space, a phase portrait occupies a single point in the parameter space. So as far as adjacency in the parameter space is concerned, we don't need to worry about the internal structure of a phase portrait.

To emphasize the common structure underlying both search algorithms, I have implemented parameter space searching by a consistency maintenance algorithm formally identical to the one used in phase space searching. Besides capturing a common pattern of behavior, this implementation has the advantage of being extendable to a two-dimensional parameter space.

### 5.5.1 Representation of a Phase Portrait

The basic object for parameter space searching is a phase portrait. Each phase portrait, as discussed in the previous chapter, is represented by a collection of canonical flow patterns, which are the output of the phase space searching module.

### 5.5.2 Representation of a collection of Phase Portraits

A data structure, called a *FAMILY*, is used to represent the list of phase portraits explored. The *FAMILY* is the basis for generating a pseudo-English summary description of the bifurcation patterns, and answering questions about them. The internal representation of *FAMILY* contains information about the range of the parameter value, the types of bifurcation patterns found, and statistics about the current simulation run.

### 5.5.3 Consistency Complaints

When the family of phase portraits is not self-consistent, a complaint will be made. A **consistency complaint** is a data structure describing the nature of the complaint. Specifically, it records the type of complaint, and the identity of the phase portraits involved. A **complaint-dispatcher** examines the type of a complaint, and invokes the appropriate suggestion rule to propose a new candidate parameter value.

### 5.5.4 Basic Algorithm

The basic data structure is a **flow pattern adjacency graph**. The graph has one type of node, and one type of link. Each node of the graph represents a flow pattern. Each link stands for an adjacency relation. We say an adjacency link between two nodes is **valid** if the orbits in question are adjacent in the parameter space. An adjacency link is **inconsistent** if the adjacency cannot be part of a legal bifurcation pattern.

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 removed from the stack when the adjacency link causing the complaint is no longer valid. KAM continually searches for new flow patterns until the complaint-stack is emptied.

The principal steps of consistency maintenance are:

1. Initially, pick some random parameter values. Create flow patterns corresponding to these values.
2. Add the newly created flow patterns to the flow pattern adjacency graph.
3. Update adjacency links. Produce a list of invalid adjacency links to be removed, and a list of new adjacency links to be added.
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.
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.
6. Handle the complaint on top of the complaint-stack. Examine the type of complaint, and propose a new parameter value to try.
7. Create a new goal with the suggested parameter value. Make a new flow pattern with this parameter value.
  - If the new goal leads to a flow pattern that is both qualitatively equivalent and very close ( $< 0.1\%$  of the parameter range) to an adjacent phase portrait, then the goal is abandoned, and the associated complaint removed.
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. Note that a consistent flow pattern adjacency graph – a graph all of whose portraits are pairwise and singly consistent – is a necessary but not sufficient condition for the self-consistency of a family of flow patterns. For instance, KAM will ignore the inconsistency between two flow patterns if the “gap” between them in parameter space is too small (see step 7 above).



## 5.6 EXPERIMENT

Let me use the Henon Map to illustrate parameter space searching using the consistency maintenance algorithm. In particular, I will start with parameter value  $A = 1.328$ , which gives rise to the familiar picture that we have seen several times before. The goal is to show how KAM systematically explores the parameter space, identifying the interesting bifurcation patterns as  $A$  varies from 0 to 2.2.

KAM explores a total of 38 phase portraits. It consumes about 120 hours of computer time. The most interesting part of the experiment is the metamorphosis of the island chain structure in the parameter range  $(1.328, 2.2)$ . Comparing KAM's findings with what's reported in Henon's paper, we discover that KAM finds something new. KAM reports that a large 7-island chain should exist in the parameter range  $(1.818, 1.846)$ . This bifurcation is missing from Henon's paper.

The experiment will be presented in three parts: first, reporting what happens from 1.328 to 1.764; second, from 1.764 to 1.982; third, from 1.982 to 2.173. In the first parameter regime, KAM finds 8 phase portraits; in the second, it finds 4; in the third, 8.

### 5.6.1 From $A = 1.328$ to 1.764

Fig. 5.11 displays the canonical flow patterns corresponding to 8 phase portraits that KAM found in this parameter regime. The following is a summary of the steps leading to the discovery of these phase portraits.

1. KAM is given  $A = 1.328$  as the initial parameter value. It finds 10 orbits in the phase portrait for this parameter value (see previous chapter), and the result is a flow-pattern-1-1 with period 5 and rotation numbers  $= ((\frac{1}{5}, \frac{1}{4}), (\frac{4}{21}, \frac{5}{26}))$ .
2. The **non-maximal-parameter rule** fires. It picks 1.764 (which is the average of 1.328 and 2.2) as the next parameter value to try, and eventually creates a flow-pattern-0-0 with rotation number  $= (\frac{4}{15}, \frac{3}{11})$ .
3. The **Poincare-Birkhoff-8 rule** fires because the two canonical flow patterns are not consistent. The reason is that their rotation numbers are not adjacent in the farey ordering (because two low-order farey fractions,  $\frac{1}{4}$  and  $\frac{1}{5}$ , are between  $\frac{5}{26}$  and  $\frac{4}{15}$ ). So KAM picks 1.546, the midpoint between 1.328 and 1.764. The result is a flow-pattern-0-0 with rotation number  $= (\frac{3}{13}, \frac{7}{30})$ .

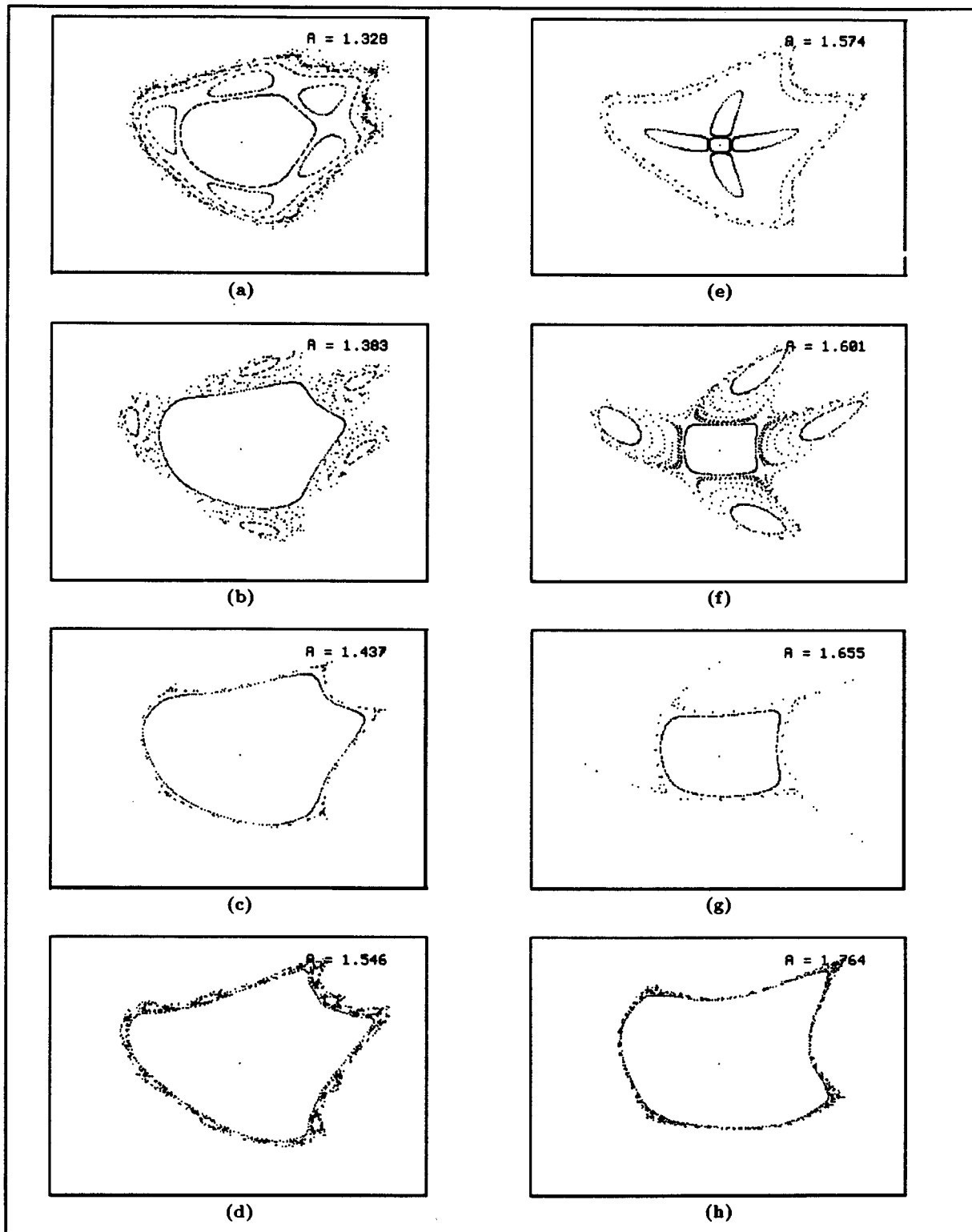


Figure 5.11: *KAM finds eight flow patterns in the parameter regime  $(1.328, 1.764)$*

4. The **Poincare-Birkhoff-8** rule fires again because the rotation numbers of the flow-pattern-1-1 with  $A = 1.328$  and flow-pattern-0-0 with  $A = 1.546$  are not farey adjacent. Also, the **qualitative-equivalence** rule fires because the the rotation numbers of two adjacent flow patterns (with  $A = 1.546$  and  $1.762$ ) with no island chain are not farey adjacent. Two inconsistency complaints are stored on stack.
5. KAM handles the first complaint. It picks  $1.437$ , the midpoint between  $1.328$  and  $1.546$ . This parameter value results in a flow-pattern-0-0 with rotation number =  $(\frac{4}{19}, \frac{3}{14})$ .
6. The **Poincare-Birkhoff-8** rule fires a third time because  $(\frac{4}{21}, \frac{5}{26})$  are not farey adjacent to  $(\frac{4}{19}, \frac{3}{14})$ . KAM picks  $1.383$ , the midpoint between  $1.328$  and  $1.437$ . This time the result is a flow-pattern-1-0 with a 5-island chain not bounded by any closed curve.
7. At this stage, the four canonical flow patterns in the range  $(1.328, 1.546)$  are consistent. So KAM will handle the inconsistency complaint that is stored on stack. KAM picks  $1.655$ , the midpoint between  $1.546$  and  $1.764$ . The result is a flow-pattern-0-0 with rotation number =  $(\frac{1}{4}, \frac{7}{27})$ .
8. The **qualitative equivalence** rule fires because the flow pattern with  $A = 1.655$  is not consistent with the one with  $A = 1.546$ . Another midpoint,  $1.601$ , is chosen. This time KAM finds a flow-pattern-1-0 with a 4-island chain and rotation number =  $(\frac{1}{4}, \frac{1}{3})$ .
9. The **Poincare-Birkhoff-9** rule fires because the flow-pattern-1-0 with  $A = 1.601$  and the flow-pattern-0-0 with  $A = 1.546$  are not consistent. So the midpoint,  $1.574$ , is tried. The result is a flow-pattern-1-1 with a bounded 4-island chain. At this stage, all the flow patterns in the range  $(1.328, 1.764)$  are pairwise consistent.

## 5.6.2 From $A = 1.764$ to $1.982$

This parameter range is most interesting because KAM finds a bifurcation that was not reported in Henon's paper. KAM reports that a 7-island chain (and hence a stable period-7 orbit) exists in the parameter range  $(1.818, 1.846)$  (Fig. 5.12). That a period-7 Poincaré-Birkhoff bifurcation should exist between the period-4 bifurcation and the phantom 3-kiss is not obvious from reading Henon's paper. In fact, what he showed seems to be a smooth change in the island chain structure as the parameter is varied

from 1.60 to 2.01 – a 6-island chain appearing and disappearing, then a 5-island chain, then a 4-island chain, and finally a 3-island chain.

The flow pattern with  $A = 1.764$  is not consistent because of the non-maximal parameter rule. The new value picked is 1.982. That results in a flow-pattern-0-0. Skipping some of the details, we note that three more flow patterns are found, at 1.873, 1.818, and 1.846, in order to make all the flow patterns in this parameter range pairwise consistent.

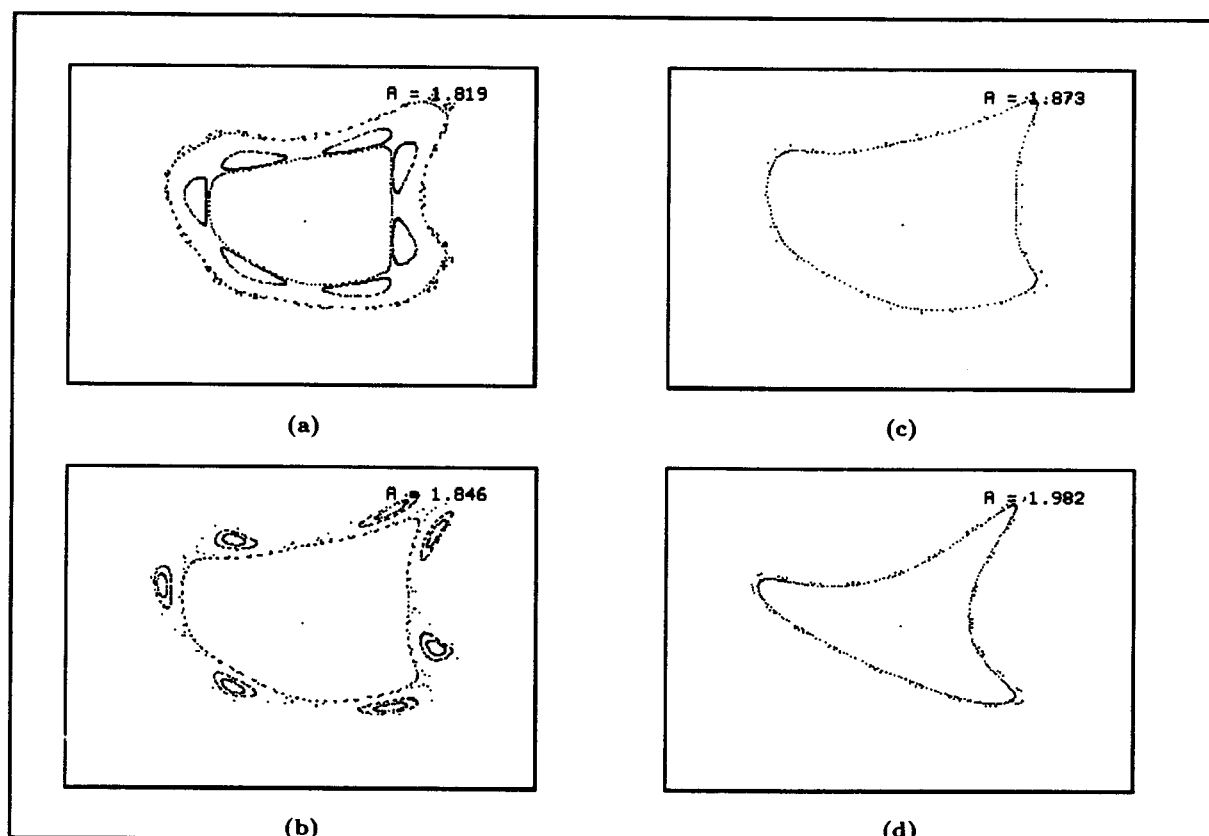


Figure 5.12: *KAM finds a large 7-island chain in the parameter regime (1.764, 1.873)*

### 5.6.3 From $A = 1.982$ to 2.173

A phantom-3-kiss bifurcation occurs in this parameter range. Fig. 5.13 shows the eight flow patterns that KAM finds. Note that at  $A = 2.064$ , the fixed point has momentarily turned unstable (Fig. 5.13f).

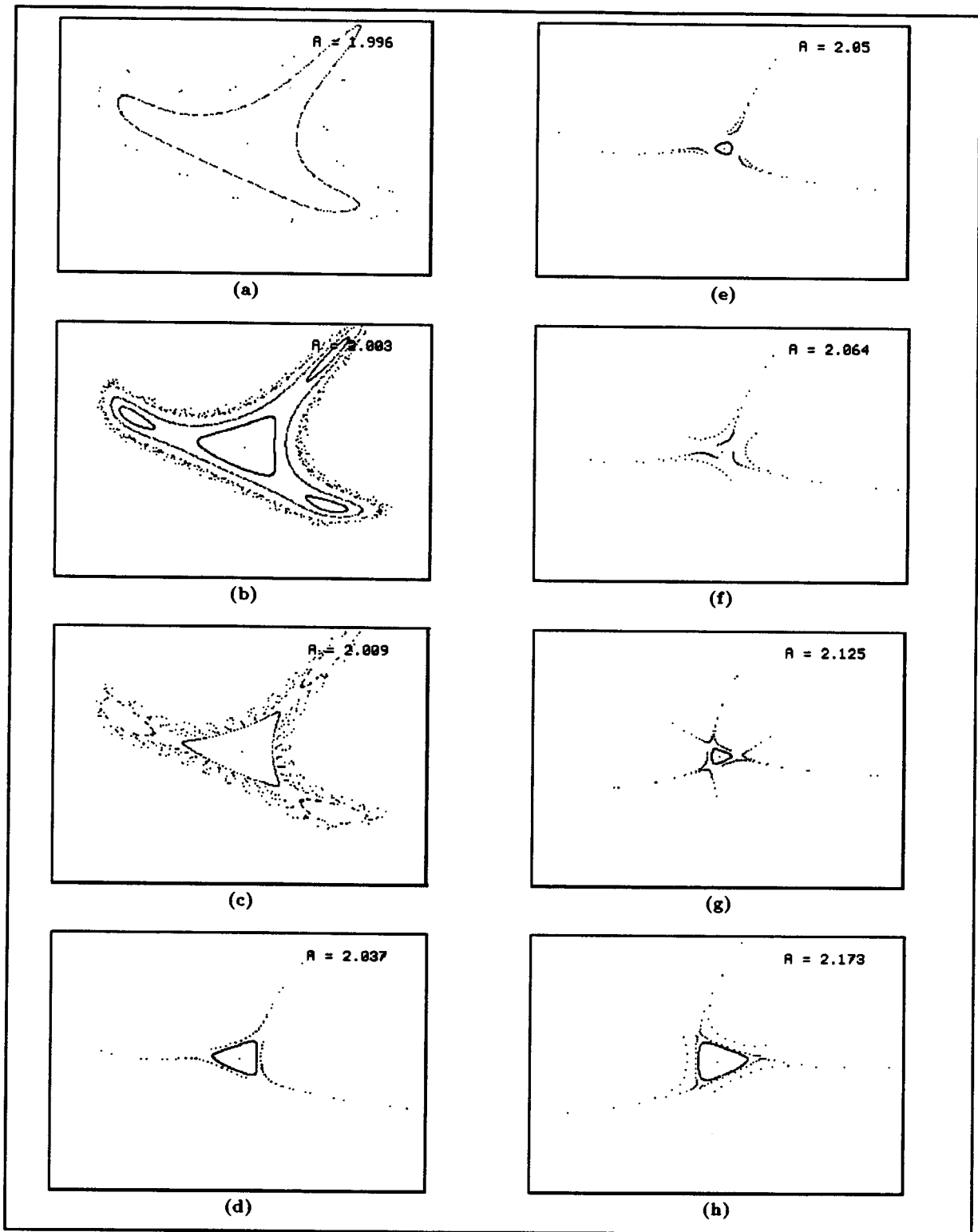


Figure 5.13: *KAM finds a phantom-3-kiss bifurcation in the parameter regime (1.982, 2.2).*

## 5.7 OTHER EXPERIMENTS

Running a complete parameter space search takes a very long time because one phase portrait can consume several hours of computer time. For instance, it takes about a week of computer time (on a Symbolics) to generate the 38 flow patterns for the Henon Map. So I have only tried one more set of extensive experiments: the Bak Map. I picked the Bak Map because it displays the Rimmer bifurcations and period-doubling, which are not found in the Henon Map.

For the Bak Map, KAM finds a total of 27 flow patterns in the parameter range (1, 4.5). Recall at 4.0, a period-doubling occurs: the fixed point bifurcates into a stable period-2 orbit.

## 5.8 EVALUATING THE PERFORMANCE

I will evaluate the parameter space search algorithm with respect to three criteria: efficiency, completeness, and robustness.

Efficiency is measured by the number of non-redundant flow patterns that the program finds. A flow pattern is redundant if its topological structure is completely identical to one already found. The smaller the amount of redundancy the more efficient the program is. With respect to this criterion, KAM performs extremely well. Out of the 38 flow patterns for the Henon Map, only 4 of them are redundant. For the Bak Map, only 1 of the 27 flow patterns is redundant.

Completeness concerns the ratio of the bifurcations that KAM discovers to the total number of "interesting" bifurcations that exist. I count only "interesting" bifurcations – bifurcations of low-order island chain and fixed point – because bifurcation theory tells us that there are always infinitely many bifurcations of small, high-order island chains. These high-order island chains usually occupy so tiny a fraction of the phase space that they can be safely ignored.

For the Henon Map, we have seen that KAM's result is better than that given in Henon's paper. KAM finds a significant Poincaré-Birkhoff period-7 bifurcation that Henon missed. By calculating the multiplier of the fixed point of the Henon Map as a function of the parameter, we can show that all the possible bifurcations of low-order

island chains are in fact discovered by KAM. So for Henon Map, KAM's search algorithm is complete. In fact, it should be mentioned that a similar calculation shows that the search algorithm is also complete with respect to the Bak Map.

Robustness is the final criteria. It asks how well the program performs if the input contains some error in it. For instance, can the program recognize that the input is erroneous?

Recognizing erroneous input is an important problem because as we have seen the phase space searching program may fail to find some island chain. As a result, a wrong canonical flow pattern will be used to represent to actual phase portrait.

Unfortunately, the parameter space search program has no way to detect such error at all; it can be "fooled" easily. In contrast, the phase space search program is more robust. If you hand it some random collection of points, the program will complain if this "orbit" intersects some other existing orbits.

What the parameter space search program can do is to help phase space searching by suggesting initial states that are likely to result in island chains. The program is able to make such suggestions whenever it sees island chains in some nearby flow patterns. Let's see an example. Suppose the same island chain is found at parameter values  $p_1$  and  $p_2$ . Then if it is required to create a phase portrait at some  $p_3 \in (p_1, p_2)$ , it will be useful to interpolate the locations of the known island chains to provide a good guess for searching the island chain at the new parameter value.

## 5.9 EXECUTIVE SUMMARY OF NUMERICAL EXPERIMENTS

In the Introduction Chapter of this dissertation, I stated that the goal of KAM is to produce a high-level qualitative description – an executive summary – from the output of numerical experiments. The summary should contain essential information about the dynamics that an engineer or a physicist studying the system may want to know. This section will describe how KAM generates such a summary.

The summary consists of three pieces of information: (1) a description of the major bifurcation patterns as some system parameter is varied, (2) a graph showing the relative area covered by regular motion as a function of the parameter, and (3) a summary picture showing the behavior of the system for initial conditions on some cross-section of the phase space for all values of the parameter. Let me describe them in more detail.

## Description of Bifurcation Patterns

In exploring the Henon Map in the parameter range  $(0, 2.2)$ , KAM discovers 38 phase portraits. Too many details are contained in these pictures to allow a dynamicist to grasp the essence of the system in a glance. So a more compact description is desirable.

Fig. 5.14 shows the summary description that KAM generates. That the description is written in some pseudo-English is not important – it is for the benefit of the person reading the output. What's important is that this information is captured in some internal formal language so that the information can be manipulated and reasoned about by other programs. For example, we may use this kind of description for comparing the overall dynamical behavior of two dynamical systems. The information may also allow a user to choose a specific parameter range for more detailed study.

In the parameter range  $(0, 2.2)$ , KAM finds a total of 38 phase portraits.  
The following bifurcation patterns are observed:

Poincare-Birkhoff bifurcation:  
A pair of period-8 orbits appears at about 0.875 and the 8-island chain disappears at about 1.075.

Poincare-Birkhoff bifurcation:  
A pair of period-7 orbits appears at about 1.000 and the 7-island chain disappears at about 1.157.

Poincare-Birkhoff bifurcation:  
A pair of period-6 orbits appears at about 1.075 and the 6-island chain disappears at about 1.257.

Poincare-Birkhoff bifurcation:  
A pair of period-5 orbits appears at about 1.271 and the 5-island chain disappears at about 1.437.

Poincare-Birkhoff bifurcation:  
A pair of period-4 orbits appears at about 1.574 and the 4-island chain disappears at about 1.655.

Poincare-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.

Figure 5.14: KAM generates a summary description of the major bifurcation patterns for the Henon Map.

The summary describes when major bifurcations occur. The most frequent bifurcation, as expected, is the Poincaré-Birkhoff . It is easy to generate the description for this type of bifurcation – only one quantity, the span of an island chain, is needed. The span of an island chain is the parameter range in which the island chain persists. For instance, the 5-island chain in the flow pattern with  $A = 1.328$  is first seen when  $A$  is 1.271, and does not appear in the flow pattern before that. The island chain is seen breaking away



from the central regular region when  $A$  is 1.383. By 1.437, the island chain has completely vanished. The values, 1.271 and 1.437, provide a conservative estimate for the span: the island chain has already been created by 1.271, and destroyed by 1.437.

The extremal bifurcation is detected in the same way: the period-3 island chain does not appear before  $A = 2.003$ , and vanishes at about 2.037.

Estimating when the phantom-3-kiss occurs is somewhat different. Here, we don't have any island chain to follow. But KAM knows that the domain of stability around a fixed point will shrink to a point when the unstable period-3 orbit collides with the fixed point. So it looks for a flow pattern in which the region surrounded by the outermost closed curve is vanishingly small. The flow pattern at 2.064 satisfies this condition.

Of course, by looking at the single flow pattern at 2.064, KAM cannot conclude that this must be a phantom-3-kiss because the fixed point can lose its stability via other types of bifurcation. To make such conclusion, KAM gathers more evidence by checking the topology of the nearby flow patterns. For instance, if the flow patterns near the one at 2.064 have a 3-island chain or if their outermost closed curves have rotation number close to  $\frac{1}{3}$ , then KAM will report detecting a phantom-3-kiss bifurcation.

### Graph showing relative area of regular region

Fig. 5.15 is a plot of the relative area of regions occupied by quasiperiodic orbits, periodic orbits and island chains as a function of the parameter. A large area at a particular parameter value means that for that parameter value the majority of initial conditions will result in regular, predictable motion. Conversely, a small area means that the phase space is mostly inhibited by chaotic orbits.

The plot shows two significant dips in the area of the regular region: first, at about 1.6; second, at about 2.0. Such information may be valuable to someone designing a system. For instance, if it is desired that the system should stay around the fixed point for a large range of initial conditions, the designer may want to avoid operating the system around the parameter value 1.6 and 2.0.

The relative area of regular region of a flow pattern is calculated by summing two quantities: the area bounded by the outermost KAM curve, and the area of any island chains not bounded by KAM curves. One data point is obtained for each flow pattern. The smooth curve in the plot is constructed by spline interpolation using all the data points.

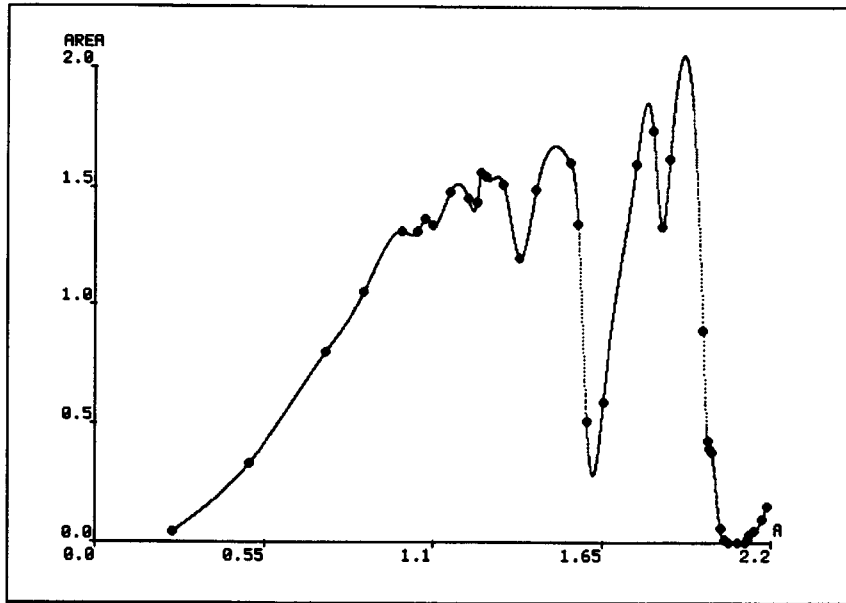


Figure 5.15: A graph showing the area of regions in phase space occupied by non-chaotic orbits as a function of parameter.

### Picture of a cross-section of the mapping

The graph of relative area is useful for some purpose, but it does not capture the effect of initial conditions as a function of the parameter. For instance, it may be interesting to know how the system behaves for a fixed initial state over a range of parameter values. Here are some of the questions we may ask. If the system starting at some state behaves regularly at some parameter value, will it become chaotic if the parameter is increased to certain value? Or, what are the states that correspond to some kind of periodic motion (i.e., hopping from islands to islands)?

To do this brute force would require an enormous amount of computation because one has to calculate an orbit for *each* initial state and *each* parameter value. We may cut the computation down by looking only at a cross-section of the initial states. We can, for instance, pick some line of initial states (preferably, an axis of symmetry), and ask what sort of orbit that will come out of it.

This is in fact what Henon did in his paper. The symmetry line he chooses is labeled 0W, which passes through the origin, and makes an angle of  $\frac{\pi}{2}$  with the horizontal axis (Fig. 5.16).

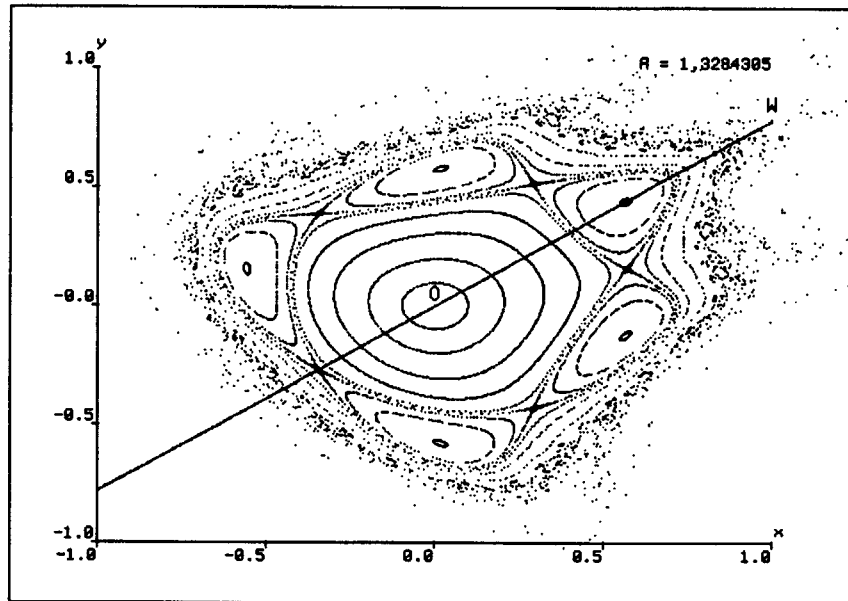


Figure 5.16: The solid line (*OW*) passing through origin and making an angle  $\frac{\Delta}{2}$  with the horizontal axis is the axis of symmetry for the Henon Map.

To generate a graph showing the type of orbit along some symmetry line, we don't need to directly compute the orbit for each point on the line. Many of these computations would be redundant. For instance, almost all <sup>3</sup> the initial states that lie inside an island chain must be hopping from island to island regularly even though we have not actually computed their orbits. This fact is a consequence of the topological constraint: closed curves will be mapped to closed curves, and the points inside can't escape.

KAM exploits this topological constraint to reduce the amount of computation. It classifies the type of a given phase space point by determining whether the point is enclosed by a closed curve, or by an island, or by neither. In the first case, the point is represented by a dot in the picture; in the second, by a circle; third, by a blank (Fig. 5.17).

---

<sup>3</sup>Not *all* because if we look at the islands in a finer scale, we'll discover that some initial states will do quite complicated things. For instance, there may be island chains inside the original islands. These smaller island chains will in turn be surrounded by secondary homoclinic orbits. Points on or near the homoclinic orbits can move in a very unpredictable manner.

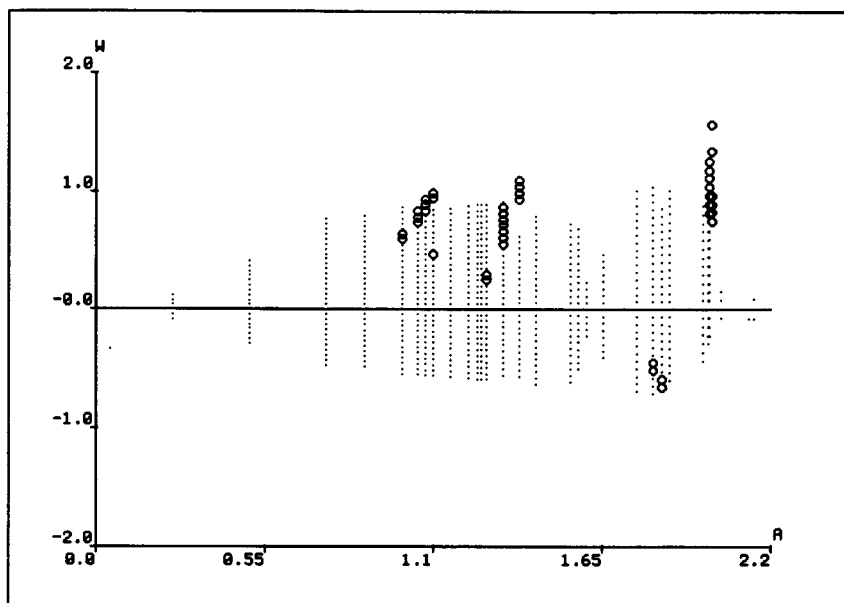


Figure 5.17: A picture showing the type of orbit for all initial points on the axis of symmetry for 38 parameter values. Dots (•) indicate quasiperiodic orbits. Circles (○) stand for islands. Blanks are for orbits that escape or are chaotic.

## 5.10 CONCLUSION: THESIS RESTATED

This chapter shows how the potentially infinite search problem of finding interesting bifurcation patterns in a one-dimensional parameter space can be effectively reduced to a simple, local consistency analysis of neighboring flow patterns analogous to that used in phase space searching. Instead of randomly trying many parameter values, KAM carefully examines each flow pattern found to see if the flow pattern is consistent with its neighbors. The consistency of flow patterns is derived from a deep mathematical theory – generic bifurcation of fixed and periodic points. A simple grammar of behavior, expressed in terms of a few dozen production rules, embodies knowledge of canonical flow patterns, and their modes of transition. KAM focuses its search in regions of parameter space where the geometry of flow patterns violates the grammar.

This chapter also provides the third example <sup>4</sup> of a problem solver whose computational structure can be nicely described by the aggregate-partition-classify paradigm.

<sup>4</sup>The other two examples are the orbit recognition program and the phase space searching program.

The aggregation step – determining adjacency of flow patterns in the parameter space – is simple here because the parameter space is restricted to one dimension. But in general the aggregation problem for multi-dimensional parameter space is still open. The partitioning step – grouping of flow patterns into larger meaningful patterns – is based on domain-specific knowledge that island chains and stability of fixed (or periodic) points are important features of dynamics. Finally, the classifying step – assignment of semantic labels to groups of flow patterns – uses the vocabulary of generic bifurcation patterns.

Last, but certainly not least, is the power of Poincaré’s idea of representing complicated dynamics by *pictures*. We have already seen how a phase portrait represents solutions to all possible initial states in a single picture. Geometry of orbits and their modes of transition provide the basis for efficient phase space searching. Here, the geometry of flow patterns and their modes of transition play the same crucial role in parameter space searching. Geometric thinking does not automatically give us answers to all interesting questions about dynamics. But it makes the problem manageable.

## Chapter 5 Summary

A bifurcation of a dynamical system is any change in the character of its steady state behavior as some or all its parameters are changed. In the language of modern dynamics, a bifurcation is any change in the topological structure of a phase portrait. Parameter space searching is the problem of (1) seeking all topologically different phase portraits in a parameter regime, and (2) interpreting the search results in terms of typical bifurcation patterns.

Efficient search for representative phase portraits is implemented by a **local consistency maintenance algorithm** analogous to the one used in phase space searching. The conditions for orbit consistency are derived from a deep mathematical theory, the generic bifurcation of fixed point of one-parameter area-preserving maps. When two incompatible flow patterns are found, the region in parameter space between the two flow patterns is searched for missing flow patterns. Resulting flow patterns are **aggregated** in the **flow pattern adjacency graph**, which keeps track of the spatial relationships among flow patterns. The search process is terminated when the adjacency graph is self-consistent. The adjacency graph is **partitioned** into groups of flow patterns in terms of bifurcations of island chains or fixed points. The bifurcations are **classified** into five generic types.

The final output of KAM is an executive summary of the entire numerical experiments. It consists of three parts: (1) an English description of the major bifurcation patterns, (2) a graph showing the relative area occupied by regular motion as a function of parameter, and (3) a picture showing the types of orbit in a cross-section of the phase space as a function of parameter.

The parameter space search program are tested with two area-preserving maps: the Henon Map, and Bak Map. KAM not only is able to automatically reproduce bifurcation patterns and their qualitative descriptions similar to those appearing in published papers, but also finds interesting bifurcation patterns that previous researchers have missed.

# Chapter 6

## Solving an Open Problem

In the previous three chapters, we have seen that by combining geometric and symbolic computation with knowledge of dynamics, it is possible to build a computer program, KAM, that automatically plans, monitors, and interprets numerical experiments with a special class of dynamical systems. The present chapter has two purposes: (1) to demonstrate the usefulness of KAM by showing how KAM was used to solve a fluid dynamics problem, a problem that I had not anticipated when KAM was designed; and (2) to illustrate how KAM might aid an engineer in the solution of a real physical problem.

The case study involves analyzing wave motion in a rectangular tank driven periodically by a wave-maker. This problem has been studied for quite some time by many fluid dynamicists [29, 15, 20], but is still not completely resolved. Using KAM's numerical findings, I was able to obtain a first insight into the character of the solution in a very short time. Once sufficient knowledge about the behaviors of the solution had been gained, I applied a well-known analytical approximation method – Chirikov's resonance overlap criterion – to the problem. The approximation was incorporated into KAM. KAM was able to make theoretical predictions about the onset of chaotic behavior that compared favorably with its own numerical results. <sup>1</sup>

This chapter is devoted to presenting a scenario that shows KAM aiding an engineer in the analysis of the wave motion problem, and explaining how the new capabilities

---

<sup>1</sup>The wave tank problem was first suggested to me by Professor Dick Yue in the Ocean Engineering Department of MIT. The entire case study, including the implementation of new capabilities, was completed in seven weeks. A paper describing the main results of this chapter has been submitted to *Journal of Fluid Mechanics*.

demonstrated in the scenario are implemented. Wave motion is a complicated piece of physics; consequently, some readers may find this chapter a little hard-going. Those readers might prefer to ignore the mathematical details in the initial reading. It should be noted that there is a rigorous mathematical basis for the informal and at times hand-waving discussion in section 6.3. Lest some readers may get lost in the details, I will give the punch line right away: *KAM does not just work for a handful of preselected problems; it captures significant knowledge about dynamics and geometry that allows it to contribute to the solution of open scientific problems.*

## 6.1 SCENARIO

The following sample dialogue shows how KAM might aid an engineer in exploring the dynamics of the wave tank. All the capabilities shown have been implemented. As in Chapter 2, the session is presented via an English dialogue solely for the benefit of the reader; KAM has no natural language capability.

**Engineer:** Here are the amplitude equations for the lowest longitudinal and transverse resonant modes. [*He types in the equations (Fig. 6.6). He also types in algebraic equations relating the coefficients in the amplitude equations with the physical parameters of the tank.*] Show me the phase portrait with the following parameter values: length-width ratio  $l = 0.248$ , water depth  $d = 1.6$ , driving amplitude  $\epsilon = 0.009$ , and detuning frequency  $\lambda = 0.2$ .

**KAM:** To plot the phase portrait, I need information to construct the Poincaré section. You can get a Poincaré section by specifying a constant energy surface, and the value of one canonical variable.

**Engineer:** Look at the energy surface  $E = 10$ . [*He types in the expression for  $E$ , the hamiltonian of the system (Fig. 6.6).*] Also, choose the Poincaré section at  $q_1 = 0$ .

**KAM:** [*After some time, KAM displays a phase portrait (Fig. 6.7), and reports its findings.*] The phase portrait has two canonical flow patterns. The first canonical flow pattern has an elliptic fixed point at  $(5.5, -1.282)$ . Surrounding the fixed point is a regular region bounded by a KAM curve with rotation number between  $\frac{1}{12}$  and  $\frac{2}{41}$ ... The second canonical flow pattern has an elliptic fixed point at  $(-5.488, 1.294)$ ... Chaotic orbits occupy 54% of the phase space.

**Engineer:** Tell me what happens to the phase portrait when the energy is decreased to  $E = 8$ .

**KAM:** [*After some time, KAM displays a phase portrait (upper portion of Fig. 6.9), and reports its findings.*] There seems to be a Rimmer fixed point bifurcation between  $E = 8$  and  $E$



= 10. Moreover, there is a large increase in the chaotic region from 0% (when  $E = 8$ ) to 54% (when  $E = 10$ ).

**Engineer:** Call this bifurcation the *critical energy phenomenon*. Tell me if the same phenomenon occurs when you halve the detuning frequency.

**KAM:** [After some time, KAM displays three phase portraits (Fig. 6.8), and reports its findings.] There seems to be no chaotic region for  $E = 1$  and  $E = 6$ . However, when  $E = 3$ , chaotic orbits occupy 22% of the phase space.

**Engineer:** Call this bifurcation the *banded energy phenomenon*. [Now the engineer wants to use an analytical approximation method to verify the numerical results.] Show me the resonance overlap diagrams corresponding to the above two cases:  $\lambda = 0.1$  and  $0.2$ .

**KAM:** For the banded energy phenomenon, the resonance overlap criterion predicts large-scale chaos when the energy is between 2.53 and 5.96. [KAM displays the first resonance overlap diagram (the top panel of Fig. 6.13).] For the critical energy phenomenon, the resonance overlap criterion predicts large-scale chaos when the energy exceeds 9.92. [KAM displays the second resonance overlap diagram (the bottom panel of Fig. 6.13).]

**Engineer:** [He is happy with the match between the analytical predictions and numerical findings. He wants to investigate the effects of changing physical parameters.] How does the detuning frequency  $\lambda$  affect the amount of chaotic region (measured in terms of the fraction of initial conditions in the phase space that lead to chaotic responses)?

**KAM:** There is a maximum response when  $\lambda$  is about 0.22. The response tapers off at both ends. [KAM displays a plot of the resonance overlap area as a function of the detuning frequency (Fig. 6.14).]

**Engineer:** How about the water depth? How does it affect the amount of chaotic response?

**KAM:** There is a maximum response when the depth  $d$  is about 1.8. When  $d$  is increased, the response falls gradually; however, when  $d$  is decreased, a sharp decrease occurs in the amount of chaotic regions. In fact, one of the resonance zones completely vanishes at  $d = 1.36$ . [KAM displays a plot of the resonance overlap area as a function of the water depth (Fig. 6.16).]

**Engineer:** This is very interesting. Can you do the same for driving amplitude? For length-width ratio? [And so on...]

What is the point of the scenario? The scenario is not about natural language understanding. Nor is it about the wave tank; a problem from celestial mechanics, for example, will serve the same purpose. My real concern is with looking at KAM from a broader

perspective – showing how a computer program like KAM can act as a useful assistant to an engineer, and demonstrating that other useful solution methods (including analytical approximation techniques) can be easily incorporated into KAM’s framework.

Let’s turn to the capabilities KAM demonstrated in the scenario:

- KAM automatically decides what initial conditions to choose in creating a phase portrait, and interprets its findings in high-level, qualitative terms.
- KAM automatically decides what parameter values to choose (the energy values in our example) in creating a family of phase portraits, and provides a high-level description of its findings.
- KAM automatically applies an approximation method, and makes predictions about the system’s behavior.
- KAM automatically generates predictions about the effect of varying a physical parameter.

The first two capabilities – phase space searching and parameter space searching – have been described in the previous few chapters and should now be familiar to the reader. I will explain how KAM applies the approximation method to make predictions. But first, a few words about the wave tank problem, and its mathematical formulation.

## 6.2 STANDING WAVES IN A RECTANGULAR TANK

One type of problem that ocean engineers are interested in is to understand and predict the effect of wave motion on floating bodies. For instance, for many engineering purposes, it is important to calculate the pressure distribution and drag coefficients on a ship hull subject to a given wave motion. The calculation requires a precise characterization of the wave motion; but waves are generally hard to describe.

The governing equation for wave evolution, as we will see shortly, is a partial differential equation (PDE) subject to highly nonlinear boundary conditions. Solving the PDE explicitly is beyond our present mathematical methods. In addition to nonlinearity, the boundary conditions, which involve the free surface height, are generally not known

in advance. That means numerical simulation of the full evolution equation will be extremely difficult. As a result, ocean engineers usually build large towing tanks just for the purpose of measuring the ship motion, pressure, and drag coefficients empirically.

Such a state of affairs is not quite satisfying: among other things, these towing tanks are very expensive to build. So there is a practical reason why engineers might want to make the predictions theoretically. But more importantly, whether we are interested in ships or not, wave evolution is a good physical problem. It encapsulates in a concrete form a far-reaching general problem: how order and chaos arise in complicated dynamical systems.

Wave evolution is a very intriguing piece of dynamics. To have any hope of making progress in the problem, we should start with a simple physical model. Imagine a deep, rectangular tank half-filled with water. Raise one end, and then gradually set it down. What will you see? Water sloshing back and forth. In the simple case, high and low water occur at the same time at opposite ends (Fig. 6.1a); in another situation, low water occurs at each end at the same time (Fig. 6.1b). For a large tank with deep bottom, the sloshing can go on for hours with waves oscillating with remarkable regularity.

A more convenient way to trigger the wave sloshing is by installing a paddle-type wave-maker at one end of the tank (Fig. 6.2). The lower edge of the paddle is hinged at the bottom of the tank. A motor controls the oscillatory motion of the paddle. By varying the angular motion of the paddle, we can generate standing waves with various amplitudes and periods.

### 6.3 MATHEMATICAL FORMULATION

With the physical setup in place, we begin our inquiry. The primary question confronting us is how to predict the character of the standing waves – their amplitude and period – in response to different controllable parameters such as the amplitude and angular frequency of the paddle, the length-width ratio of the tank, water depth, and so on.

The first step of our inquiry is the formulation of the problem in mathematical terms. Here we must exercise our judgment: deciding what approximations or idealizations to make without losing the essence of the problem. The ability to make useful approximations – deciding what features are relevant – is probably the most valuable asset a good scientist can have.

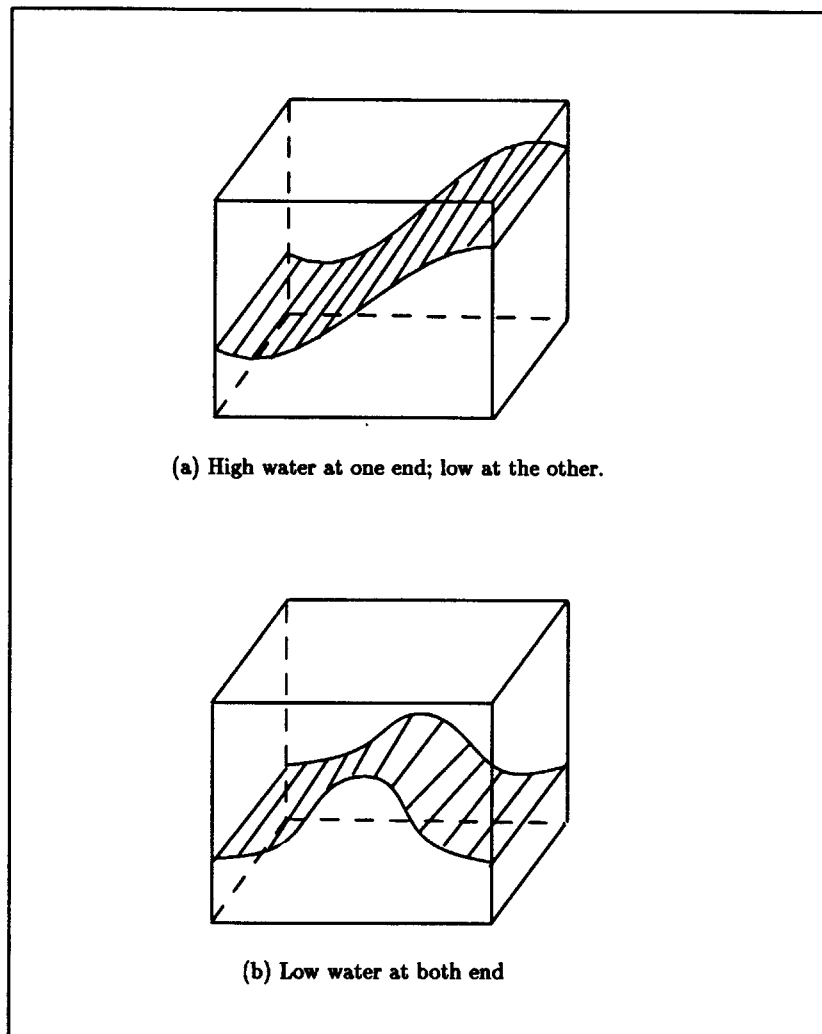


Figure 6.1: *Two sloshing modes in a rectangular tank half-filled with water. Only longitudinal waves are excited.*

So let's begin by making a few assumptions. First, we assume the motor delivers a simple sinusoidal with frequency  $\omega$  and amplitude  $\epsilon$ , the only external disturbance that upsets the equilibrium free surface. Second, we ignore surface tension and Coriolis force, pretending that gravity is the only restoring force. Waves driven by a balance of fluid inertia and its tendency under gravity to return to a state of equilibrium are called **gravity waves**. Third, we will assume the wave tank is filled with an incompressible fluid: say, water. And finally, we neglect the dissipation due to friction at the tank bottom. This is not a severe restriction because it is known that dissipation has an

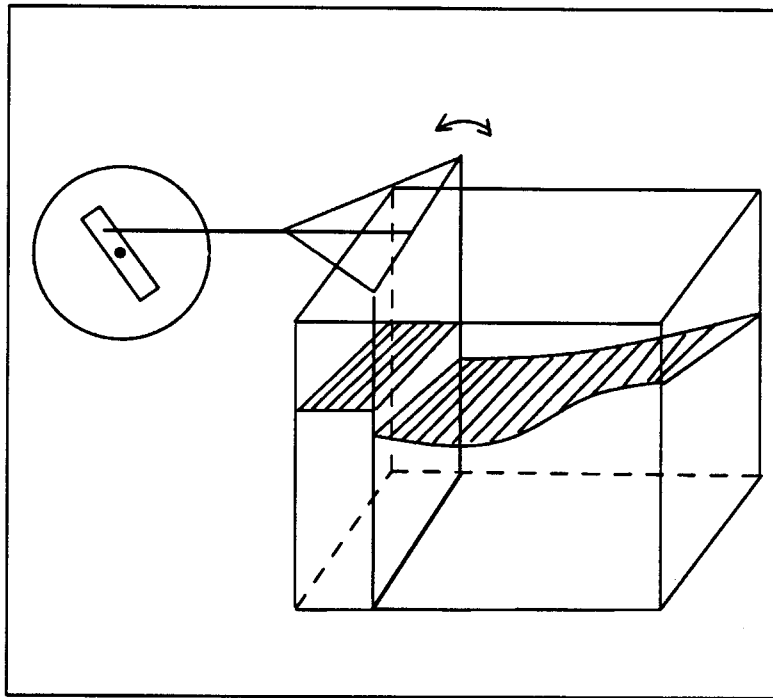


Figure 6.2: A rectangular wave tank. One end of the tank is a wave paddle driven periodically by a motor. Depending on the driving amplitude and frequency of the motor, the longitudinal and cross waves can be excited simultaneously.

extremely small effect on water waves whose wavelengths are shorter than half of the water depth. Such waves are called deep water waves.

Gravity waves have an important property that makes them easier to deal with mathematically: the fluid motion is **irrotational**. That means, throughout the fluid there is no vorticity. **Vorticity** is defined as the curl of the velocity vector; it measures the rate of rotation of a fluid particle. Vorticity is a tricky concept to understand. A fluid motion can follow a curved path and have no vorticity; on the other hand, it can follow a rectilinear path and have large amounts of vorticity. In general, a fluid motion is irrotational if it is the result of conservative forces such as gravity.

Irrotationality means the curl of the velocity vector is identically zero. So the assumption of irrotational fluid implies the existence of a scalar function, the **velocity potential**  $\phi$  such that:

$$\mathbf{V} = \nabla\phi$$

With further assumption of no friction and incompressibility, the equations of motion can be written as a Laplace equation (Fig. 6.3).

A rectangular tank of water of width  $W$ , length  $L$ , and filled to a depth  $D$ . The flow is assumed to be incompressible, inviscid, and irrotational. In terms of dimensionless variables, the governing equation is given by:

$$\nabla^2 \phi = 0$$

where  $\phi$  is the velocity potential.  
There are 5 boundary conditions:

(1) A fluid particle at the free surface ( $z = \xi(x, y, t)$ ) always remains at the surface:

$$\phi_z = \xi_t + \phi_x \xi_x + \phi_y \xi_y$$

where the subscripts denote partial derivatives.

(2) The pressure within the fluid motion must conform to Bernoulli's equation:

$$\phi_t + \xi + \nabla \phi \cdot \nabla \phi = 0$$

(3) On the sidewalls and bottom of the tank, there must be no normal velocity:

$$\begin{aligned} \phi_x &= 0 \text{ on } x = l \\ \phi_y &= 0 \text{ on } y = 0, 1 \\ \phi_z &= 0 \text{ on } z = -d \end{aligned}$$

where the tank width is scaled to 1; length, 1; and depth,  $d$ .

(4) The wave-maker oscillates with frequency  $\omega$  and amplitude  $\epsilon$ :

$$x = \epsilon \cos \omega t$$

(5) The mean surface height is taken to be  $z = 0$ .

Figure 6.3: *Formulation of the standing wave problem in a rectangular tank. It is not necessary to understand the details of the mathematical equations for an appreciation of the rest of this chapter.*

The nonlinear boundary conditions almost certainly prevent us from obtaining an explicit solution to the wave tank problem. You may try to solve the equations by brute force; that is, compute the solution numerically. But, as mentioned earlier, this method too is ineffective because of the unknown free surface conditions.

The traditional treatment of the gravity wave problem goes something like this: we do not ask for a general solution; it may be too hard to find. Instead, we concentrate our

attention on *periodic solutions* with the object of obtaining in the first place information of a *qualitative* character about the periodic solutions. Once this has been done, we'll proceed to refine the solution to obtain more accurate quantitative information.

There are several methods to obtain periodic solutions. These methods differ in their assumptions about the general character of the solution, and in the amount of computation required to obtain a reasonable approximation. The method of **multiple time scales**, which I will explain shortly, turns out to be quite useful for our specific problem. To apply the method, we need to understand the physics of the situation a bit more.

Suppose the driving amplitude of the wave-maker is small. We should expect a nice harmonic response along the length of the tank: a sinusoidal standing wave with the same frequency as that of the wave-maker. Such a wave is called a **longitudinal wave**; its crests are parallel to the wave-maker. The amplitude of the longitudinal wave depends on the **detuning frequency**, which is the difference between the driving frequency and the natural frequency of the wave. For certain detuning frequencies, the longitudinal wave achieves a maximum elevation, in which case we say the longitudinal wave resonates with the wave-maker. In physics, this type of excitation, where a vibrating element (a fluid particle in our wave example) and the external drive move on parallel planes, is called a **forced resonance** (Fig. 6.4).

Is a longitudinal wave the only kind of wave that the wave-maker can excite? The answer is no. Experimentally, it has been observed that a **cross wave**, one whose crests are at right angles to the wave-maker with half the frequency of the wave-maker can also be excited. Another name for this is the **transverse wave**, since the vibrating element moves in a direction perpendicular to that of the drive. Experiments have also shown that the amplitude of the cross wave depends on the length-width ratio of the tank.

It is not hard to explain why a cross wave can be excited. Recall one of the most favorite demonstrations in freshman physics: an inverted pendulum, which is unstable, can be converted into a stable one if a vertical pulsating force of proper amplitude and frequency is applied at the support of the pendulum (Fig. 6.4). Analogously, we can destabilize the otherwise stable downward equilibrium position of the pendulum by the same kind of pulsating force. The amazing feature of this phenomenon is that the direction of the pulsating drive is *orthogonal* to that of the pendulum's oscillatory motion. Physicists give such excitation a name, **parametric resonance**; this is so-called because a system can be set into resonant oscillation by periodically varying certain system parameters (the length of our pendulum, for example) (Fig. 6.4).

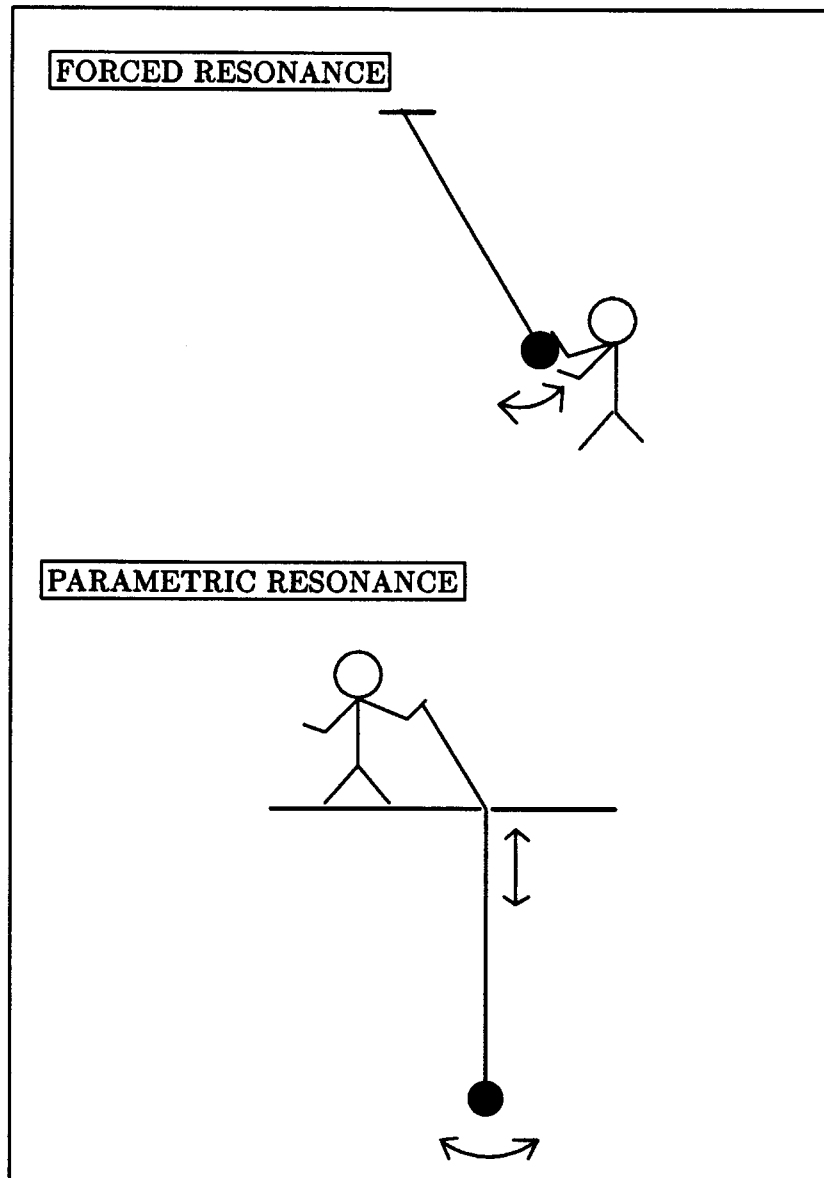


Figure 6.4: *Upper: A simple pendulum swings with large amplitude if the bob is forced periodically at the same frequency as its natural frequency. Lower: A simple pendulum can be parametrically excited if its length is varied periodically at twice its natural frequency. If the amplitude of the variation is too small, the bob just moves up and down vertically; but if the amplitude is large, the bob begins to swing spontaneously.*



We can explain how the pendulum oscillation is maintained by a simple energy consideration. The vertical pulsating force provides the pendulum with a constant source of energy. The tension of the light rod, which connects the pendulum bob to a pivot, varies cyclically as it swings through the vertical; the tension is maximum at the vertical and reaches a minimum at either end of the swing. If the periodic drive is such that the upward velocity of the pivot is in phase with the variations of the tension, then energy is fed into the pendulum at all phases of the oscillation. And since the tension runs through two cycles for each complete cycle of the pendulum, the pendulum oscillates at half the frequency of the periodic drive.

If the equations of motion for the gravity wave problem *were* linear, then any physicist worthy of his trade would be able to write down a complete solution: calculate the amplitude and frequency of the longitudinal wave, and the same for the cross wave; add the two solutions together; and we'd be home and dry. But the equations aren't linear. So we need some more fiddling.

What's the effect of the nonlinearity on the waves? Usually it means some coupling between the longitudinal and the cross waves; consequently, energy transfer between the two modes becomes possible. Now, if we assume the system is only weakly nonlinear (say, by taking the driving amplitude  $\epsilon$  to be small), then we expect that the longitudinal wave is essentially an oscillation with frequency  $\omega$  of the drive, but with *slowly varying amplitude and phase*. Likewise the cross wave is essentially an oscillation with frequency  $\frac{\omega}{2}$  modulated by a slowly varying amplitude and phase.

This idea of slow modulation of the basic frequencies is the key ingredient in applying the method of multiple time scale to our problem; that is, the wave motion is governed by two time scales: a fast time scale,  $t$ , for the basic oscillation, and a slow time scale,  $\tau$ , for the modulation (Fig. 6.5). So we can assume an approximate solution of the form:

$$\xi_0(x, y, t, \tau) = A(\tau) \cos(\omega t) \cos(\kappa_1 x) + B(\tau) \cos\left(\frac{\omega}{2} t\right) \cos(\kappa_2 y) \quad (6.1)$$

where the  $\kappa$ 's are wave numbers, and the functions  $A(\tau)$  and  $B(\tau)$  are assumed to be slowly varying relative to  $\cos \omega t$ . To account for possible phase differences between the waves and the drive, it is conventional to let  $A(\tau)$  and  $B(\tau)$  be complex functions:

$$\begin{aligned} A(\tau) &= p_1(\tau) + i q_1(\tau) \\ B(\tau) &= p_2(\tau) + i q_2(\tau) \end{aligned}$$

where the  $p$ 's and  $q$ 's are real functions.  $A(\tau)$  and  $B(\tau)$  are often called the **complex amplitudes** of the oscillations; they contain both the amplitude and phase information about the oscillations. Substituting equation 6.1 into the governing equations in

Fig. 6.3 and applying solvability conditions, we arrive at a set of four ordinary equations describing the evolution of the complex amplitudes (see Fig. 6.6).

Let's see what the amplitude equations mean. The equilibrium points of the equations occur when  $\dot{p}_1 = \dot{q}_1 = \dot{p}_2 = \dot{q}_2 = 0$ , which implies the complex amplitudes  $A(\tau)$  and  $B(\tau)$  are constants; that means we recover the solution to the linearized problem, a linear combination of two periodic solutions. Periodic solutions of the amplitude equations, however, do not necessarily imply the periodicity of each component solution; that is, neither  $A(\tau) \cos(\omega t) \cos(\kappa_1 x)$  nor  $B(\tau) \cos(\frac{\omega}{2} t) \cos(\kappa_2 y)$  is periodic unless the complex amplitudes have frequencies commensurate with  $\omega$ . Lastly, irregular solutions of the amplitude equations mean  $\xi_0$  will vary irregularly with time.

**METHOD OF TWO TIME SCALES**

**Setup:**  
A boundary-value problem depending on a small parameter  $\epsilon$ . The solution is denoted by  $\xi$ , which is a function of space and time variables.

**Step 1:**  
Let  $t$  be the fast time scale of  $O(\epsilon)$ , and  $\tau = \epsilon t$  be the slow time scale.  
Look for solution of the following form:

$$\xi(x, y, t, \tau) = \xi_0(x, y, t, \tau) + \epsilon \xi_1(x, y, t, \tau) + \epsilon^2 \xi_2(x, y, t, \tau) + \dots$$

with undetermined coefficients  $\xi_i$ 's.

**Step 2:**  
Substitute the assumed series into the governing equations to obtain a sequence of partial differential equations, one for each order of  $\epsilon$ . The undetermined coefficients are obtained by solving these PDEs subject to boundary conditions of the original problem, and solvability conditions for the PDEs.

Figure 6.5: The method of two time scales is used to determine an approximate solution to the wave tank problem. We will be interested in the first-order approximation  $\xi_0$ .

In approximating the free surface solution by a combination of two basic frequencies, we are neglecting the contributions from high frequency harmonics. But the payoff of the approximation is enormous: instead of a PDE with complicated nonlinear boundary conditions, we reduce the problem to four ODEs (ordinary differential equations) governing the evolution of the two complex amplitudes. Moreover, as a consequence of the zero-friction and incompressibility assumptions, the ODEs are canonical equations derived from a single scalar function of four variables, the hamiltonian  $H$  (Fig. 6.6).

### AMPLITUDE EQUATIONS

$$\begin{aligned}
 \dot{p}_1 &= \frac{1}{\mu} (-\gamma_1 q_1 + \delta + q_1(\Gamma_1(p_1^2 + q_1^2) + \Sigma(p_2^2 + q_2^2))) \\
 \dot{q}_1 &= -\frac{1}{\mu} (\gamma_1 p_1 - p_1(\Gamma_1(p_1^2 + q_1^2) + \Sigma(p_2^2 + q_2^2))) \\
 \dot{p}_2 &= \frac{1}{\mu} (-\gamma_2 + \beta)q_2 + q_2(\Gamma_2(p_2^2 + q_2^2) + \Sigma(p_1^2 + q_1^2)) \\
 \dot{q}_2 &= -\frac{1}{\mu} (-\gamma_2 - \beta)p_2 - p_2(\Gamma_2(p_2^2 + q_2^2) + \Sigma(p_1^2 + q_1^2))
 \end{aligned}$$

The coefficients  $\beta, \delta, \Sigma, \mu, \Gamma_i$  and  $\gamma_i$  are related to the physical parameters  $d, l, \epsilon$  and  $\omega$  by complicated algebraic equations.

The amplitude equations can be derived from a scalar function, the hamiltonian:

$$\begin{aligned}
 H(p_1, q_1, p_2, q_2) &= -\frac{1}{\mu} \left( \delta q_1 - \frac{1}{2} \gamma_1 (p_1^2 + q_1^2) + \frac{1}{4} \Gamma_1 (p_1^2 + q_1^2)^2 + \frac{1}{2} \beta (p_2^2 - q_2^2) \right. \\
 &\quad \left. - \frac{1}{2} \gamma_2 (p_2^2 + q_2^2) + \frac{1}{4} \Gamma_2 (p_2^2 + q_2^2)^2 + \frac{1}{2} \Sigma (p_1^2 + q_1^2) (p_2^2 + q_2^2) \right)
 \end{aligned}$$

Figure 6.6: *The evolution equations for the complex amplitudes of the lowest resonant modes.  $p_1$  and  $q_1$  are the real and imaginary parts of the complex amplitude for the longitudinal wave;  $p_2$  and  $q_2$ , the cross wave. These equations are nonlinearly coupled; that means there are interactions between the longitudinal and the cross waves even at the lowest resonant modes.*

## 6.4 NUMERICAL FINDINGS

### 6.4.1 Making a Poincaré section

The amplitude equations (Fig. 6.6) can be thought of as a hamiltonian system with two degrees of freedom. A complete description of the system therefore requires a 4-dimensional phase space. Flows in 4-dimensional space, however, are not easy to visualize; it is useful to find a simpler representation. Over 100 years ago, Poincaré invented the **surface of section** – a technique which bears his name – for this purpose.

The idea of the Poincaré section is quite simple (section 3.1). If  $\Sigma_E$  is a codimension one surface (i.e., one dimension less than that of the phase space) transverse to a flow, then the sequence  $x_i$  of successive intersections of an orbit with  $\Sigma_E$  provides a lot of information about that orbit. For example, if  $x_i$  is periodic, the corresponding orbit is also periodic. In particular, a fixed point on  $\Sigma_E$  corresponds to a periodic orbit with period one. Moreover, on the surface of section  $\Sigma_E$ , successive points belonging to a

quasiperiodic orbit will appear to lie on some smooth curve, while those belonging to a chaotic orbit appear to fill a region.

We can construct a surface of section  $\Sigma_E$  for a Hamiltonian system with two degrees of freedom as follows. Since the hamiltonian is a constant of motion, the 4-dimensional phase space is foliated into 3-dimensional surfaces parameterized by the value of the hamiltonian. Now instead of following the entire continuous orbit on some constant hamiltonian surface, one samples the orbit at discrete time intervals. The sampling can be triggered, for instance, whenever the orbit crosses  $q_1 = c$  for some constant  $c$ . For a given value of hamiltonian  $H = E$ , we can then solve for one of the conjugate variables  $p_1$  in terms of the other:

$$p_1 = p_1(c, p_2, q_2, E)$$

The conditions  $H = E$  and  $q_1 = c$  define a family of 2-dimensional surfaces in the phase space because there may be multiple roots for  $p_1$ . To get a single surface of section, we can pick the one with say the largest root of  $p_1$ . Call this section  $\Sigma_E$ . As the phase flow evolves with time, it repeatedly pierces  $\Sigma_E$ . The successive intersections of an orbit with  $\Sigma_E$  generates a two-dimensional discrete map  $T$ :

$$(q_2^{n+1}, p_2^{n+1}) = T(q_2^n, p_2^n)$$

where the superscript  $n$  stands for the  $n$ th intersection. The mapping  $T$  is called **Poincaré map** or the **first return map**.  $T$  is area-preserving, which is a consequence of the conservation of the hamiltonian. In general, it is difficult to obtain an explicit form for the Poincaré map given the hamiltonian equations; therefore, we have to numerically approximate it.

## 6.4.2 Compiling a description of Poincaré section into numerical procedures

KAM automatically translates a high-level description of the Poincaré section into numerical routines that compute the Poincaré map. The translation is done by a higher-order procedure, `MAKE-POINCARÉ-MAP`. The procedure takes three arguments: (1) the state equations, which are given by a system of ODEs; (2) an equation of the form  $x_j = c$ , where  $x_j$  is a state variable and  $c$  is some constant; and (3) an equation for the energy surface. It returns a procedure, `NUMERICAL-POINCARÉ-MAP`, which takes as arguments a point on the Poincaré section and a value for the energy surface, and returns another point on the Poincaré section. `NUMERICAL-POINCARÉ-MAP` numerically integrates the

state vector until it crosses the surface  $x_j = c$ . The Bulirsch-Stoer algorithm with relative accuracy  $10^{-7}$  was used to perform the integration.

### 6.4.3 Two interesting findings: banded and critical energy phenomena

Fig. 6.7 shows an example of a Poincaré section. The axes of the figure are the real and imaginary part of the complex amplitude of the cross wave. The phase plane reveals three fixed points: two elliptic fixed points surrounded by invariant closed curves, and one hyperbolic fixed point near the origin. Outside the two regular regions around the elliptic fixed points is a large chaotic zone. The chaotic zone is in turn bounded by quasiperiodic orbits. The existence of chaotic solutions, as we have seen in section 6.3, implies that the free surface waves behave irregularly as a function of time (provided that  $\xi_0$  (given by equation 6.1) is a good approximation to the actual waves).

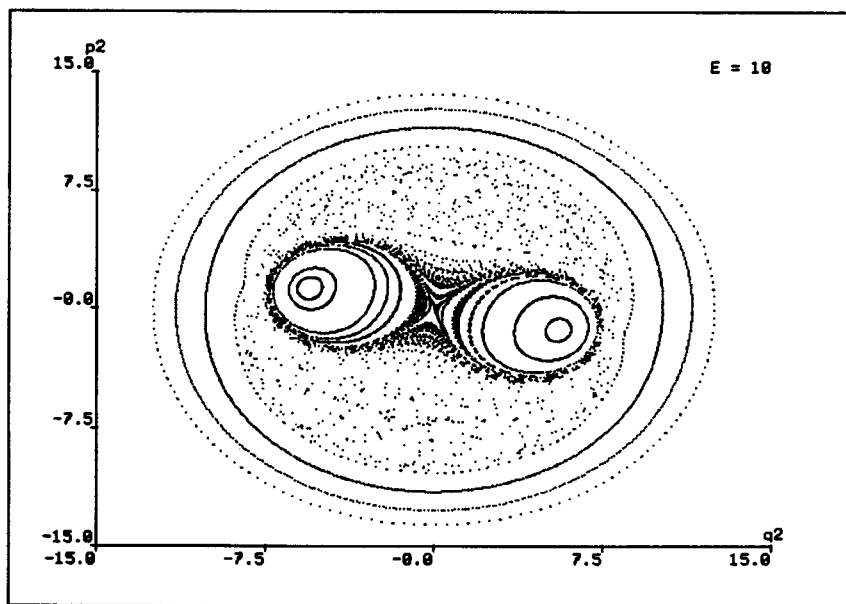


Figure 6.7: *Poincaré Map for the wave tank hamiltonian:  $E = 10$  and  $\lambda = 0.2$ . A large portion of the phase portrait is occupied by chaotic orbits.*

Numerical experiments with the amplitude equations reveal two interesting findings. First, at detuning frequency  $\lambda = 0.1$  and low energy level  $E = 1$ , the phase portrait

appears completely regular: an elliptic fixed point at the origin surrounded by a nested sequence of invariant curves (see Fig. 6.8a). When the energy is raised to  $E = 3$ , the outermost invariant curves are destroyed, and a large chaotic zone is seen (Fig. 6.8b). As the energy is further increased to  $E = 6$ , the total energy surface shrinks but the orbits appear completely regular again (Fig. 6.8c). I call this scenario the **banded energy phenomenon** because chaotic orbits seem to appear only for an interval (or a band) of energy values.

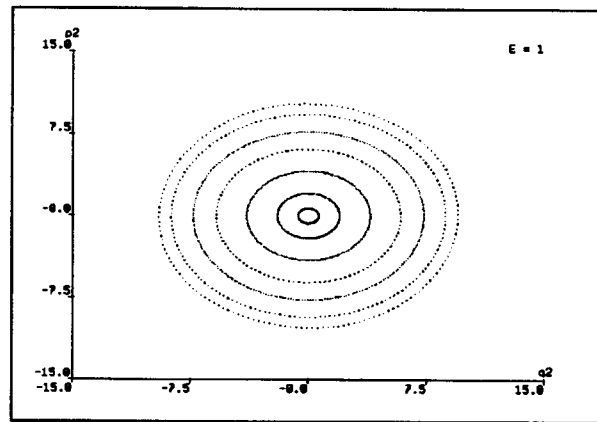
The second finding is what I refer to as the **critical energy phenomenon** because there seems to a critical energy level beyond which chaotic orbits dominate the phase space. This occurs when the detuning frequency is doubled to  $\lambda = 0.2$ . When the energy is low  $0 < E \leq 8$ , the phase portrait consists only of regular orbits (Fig. 6.9a). For energy somewhat larger, say,  $E = 10$ , we see that the elliptic fixed point at the origin has lost its stability, becoming hyperbolic and giving rise to two new elliptic fixed points (Fig. 6.9b). Moreover, a large chaotic zone occupies most of the energy surface.

## 6.5 APPLYING AN ANALYTICAL APPROXIMATION METHOD

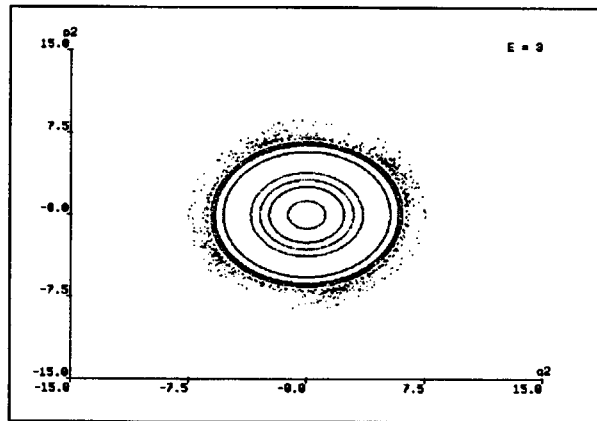
### 6.5.1 The idea of Resonance Overlap

According to the KAM Theorem, sufficiently incommensurate invariant curves persist under small perturbation of an integrable system (section 3.1). As the perturbation strength increases, neighboring resonance zones will interact. Initially, chaotic motion is confined to a narrow region around the separatrices bounding the resonance zones. As two resonance zones grow and eventually “overlap”, any invariant curves between them will be destroyed; there is no barrier to prevent orbits from diffusing throughout both resonance zones, marking the onset of large-scale chaos. The idea that the overlapping of resonances leads to widespread chaotic motion is developed by Chirikov into a practical method for estimating critical parameters governing the appearance of chaos [5]. Although resonance overlap is not a rigorous method, it is easily applicable to fairly complicated systems and extendable to higher dimensions.

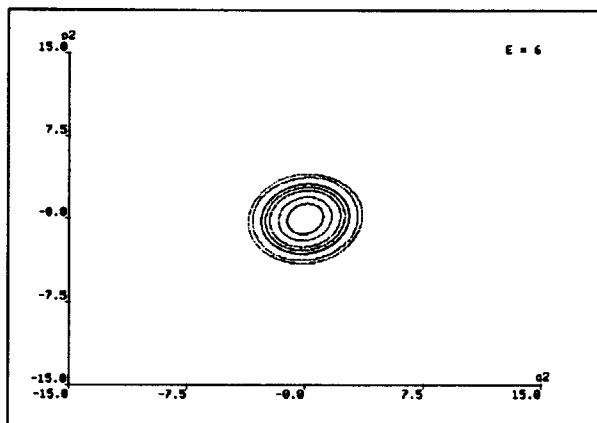
In its simplest form, Chirikov’s Resonance Overlap Criterion postulates that the last invariant curve between two lowest-order resonances is destroyed when the sum of the half-widths of the two resonance zones just equals the distance between the resonance centers (Fig. 6.10). A major assumption in the calculation is that the half-widths of



(a) Hamiltonian  $E = 1$ : regular motion dominates

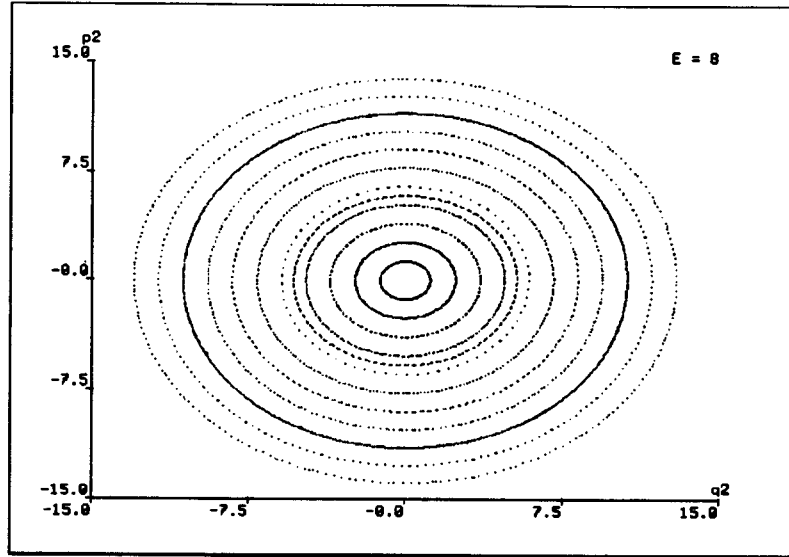


(b) Hamiltonian  $E = 3$ : widespread chaotic motion

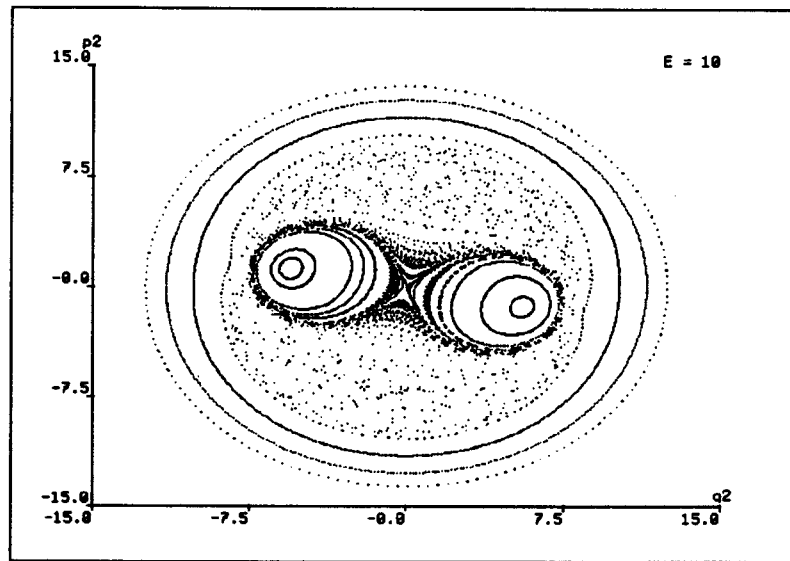


(c) Hamiltonian  $E = 6$ : regular motion dominates

Figure 6.8:  $\lambda = 0.1$ . *Banded energy phenomenon: large chaotic zones appear only for an interval (or a band) of energy values.*



(a) Hamiltonian  $E = 8$ : regular motion dominates



(b) Hamiltonian  $E = 10$ : chaotic motion dominates

Figure 6.9:  $\lambda = 0.2$ . Critical energy phenomenon: large chaotic zones appear when the hamiltonian exceeds a critical value.



each resonance zone can be calculated independent of the other. This simple criterion results in a conservative estimate because chaotic motion can result from interaction of the secondary resonances lying between the two primary resonances before the primary resonance zones touch. Nevertheless the simple criterion can quickly yield a rough estimate for the transition to large-scale chaos. The results of applying the resonance overlap calculation to the wave tank problem are shown in Fig. 6.11.

## 6.5.2 Resonance overlap diagram

The resonance overlap criterion can give us an analytical expression predicting when large-scale chaos occur. However, it is much more illuminating to apply the method geometrically. A **resonance overlap diagram** is a graphical device for performing the geometric computation.

A resonance overlap diagram has four parts: (1) the center and boundaries of the first resonance zone, (2) the center and boundaries of the second resonance zone, (3) the region where the two resonances overlap, and (4) level curves of the zero-order hamiltonian  $H_0$ . These four parts are illustrated in Fig. 6.12 for the case when the detuning frequency  $\lambda$  is 0.1.

What are the axes of the resonance overlap diagram? I have chosen to use  $I_1$  and  $I_2$ ; they are the magnitudes of the complex amplitudes  $A(\tau)$  and  $B(\tau)$ . In classical mechanics, the  $I_i$ s are called the **action variables**, or simply **actions**. But the choice of variables is not obvious. Changing to a “right” set of variables is a powerful method for solving many mathematical problems; unfortunately, there is no general recipe to guide the selection of variables.

Each resonance zone is represented by a trio of lines: the center line is the set of locations of the resonance center, and the two branches are the boundaries of the resonance zone. The two resonance zones are displayed in Fig. 6.12a and 6.12b. Superposing these two zones, we obtain the overlap region as shown in Fig. 6.12c. The level curves of the hamiltonian turn out to be parabolas, and they are presented in Fig. 6.12d. Putting all these parts together, we get the resonance overlap diagram (Fig. 6.12e).

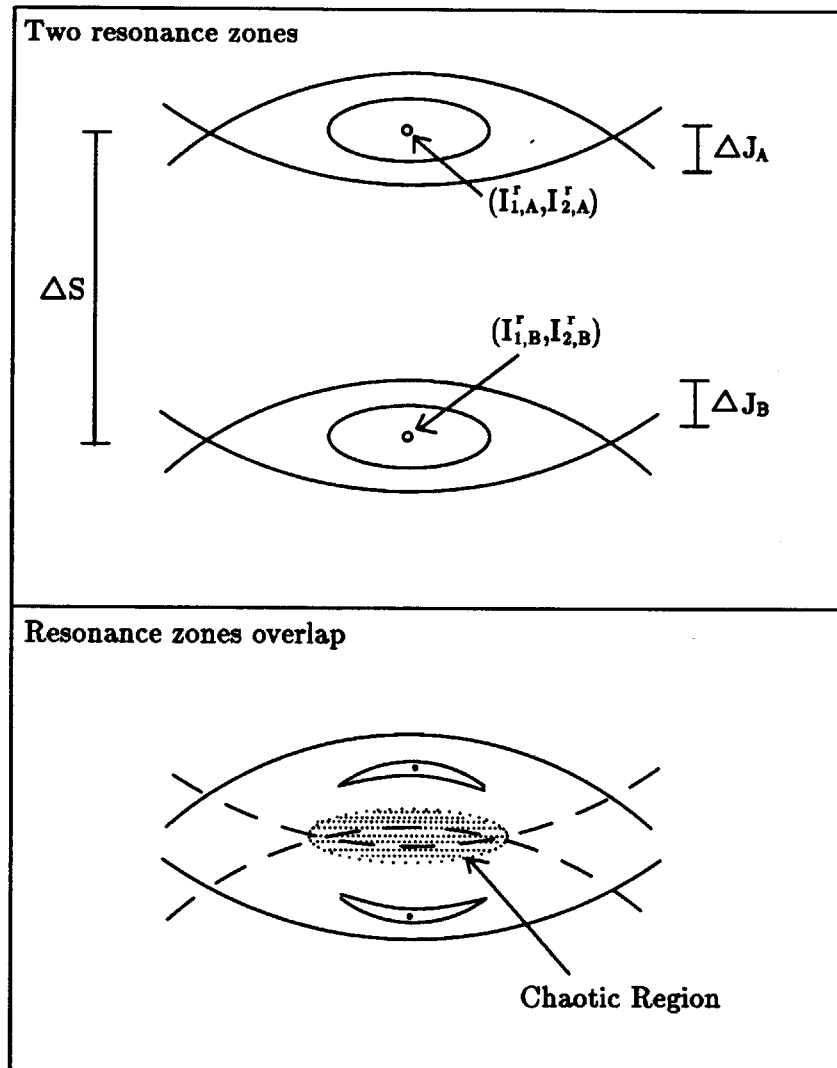


Figure 6.10: *Graphical illustration of resonance overlap criterion. When the separation  $\Delta S$  between the two resonance centers  $(I_{1,A}^r, I_{2,A}^r)$  and  $(I_{1,B}^r, I_{2,B}^r)$  is less than the sum of the half resonance widths  $\Delta J_A + \Delta J_B$ , chaotic orbits appear in the overlap area (shown as the dotted region).*

### 6.5.3 Geometric algorithms to manipulate the resonance overlap diagram

The resonance overlap diagram contains important information about the behavior of the system; for example, the critical energy levels and the extent of chaotic regions can

**APPLYING CHIRIKOV'S RESONANCE OVERLAP CRITERION**

**Step 1: Rewrite the hamiltonian in action-angle variables**

$$H = -\frac{1}{\mu} (-\gamma_1 I_1 + \Gamma_1 I_1^2 - \gamma_2 I_2 + \Gamma_2 I_2^2 + 2\Sigma I_1 I_2 - \beta I_2 \cos 2\theta_2 + \delta \sqrt{2I_1} \cos \theta_1)$$

**Step 2: Isolate the major resonances**

$$H_0 = -\frac{1}{\mu} (-\gamma_1 I_1 + \Gamma_1 I_1^2 - \gamma_2 I_2 + \Gamma_2 I_2^2 + 2\Sigma I_1 I_2)$$

$$H_A = H_0 - \frac{\beta}{\mu} I_2 \cos 2\theta_2$$

$$H_B = H_0 + \frac{\delta}{\mu} \sqrt{2I_1} \cos \theta_1$$

**Step 3: Determine resonant actions**

The resonance condition for  $H_A$  is given by:

$$I_2^r = \frac{\gamma_2 - 2\Sigma I_1^r}{2\Gamma_2}$$

For  $H_B$ , it is:

$$I_1^r = \frac{\gamma_1 - 2\Gamma_1 I_1^r}{2\Sigma}$$

**Step 4: Transform to canonical pendulum hamiltonian and determine resonance half-widths**

The resonance half width for  $H_A$  is given by:

$$\Delta J_2 = \sqrt{\frac{-\beta}{2\Gamma_2}} (I_2^r)^{\frac{1}{2}}$$

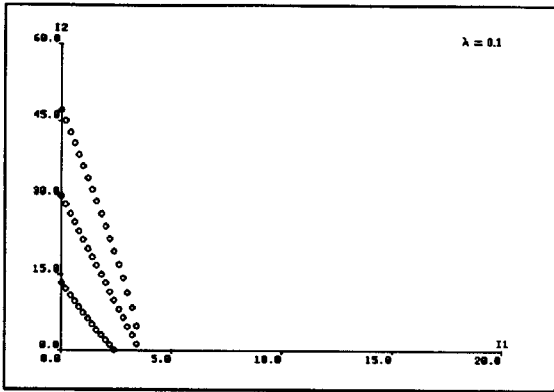
For  $H_B$  it is:

$$\Delta J_1 = \sqrt{\frac{2\sqrt{2}\delta}{\Gamma_1}} (I_1^r)^{\frac{1}{4}}$$

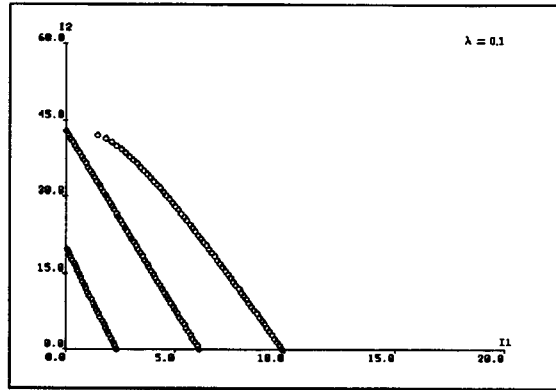
Figure 6.11: *Details of applying the resonance overlap criterion to derive the resonant actions and resonance half widths.*

be read off from the diagram. KAM automatically constructs the resonance overlap diagrams, and extracts information from them by using several well-known geometric algorithms. The main idea is to approximate each resonance zone by a convex polygon, and represent the overlap region as the intersection of two convex polygons. From such representation, it is easy to deduce the critical energy levels and the size of overlap region.

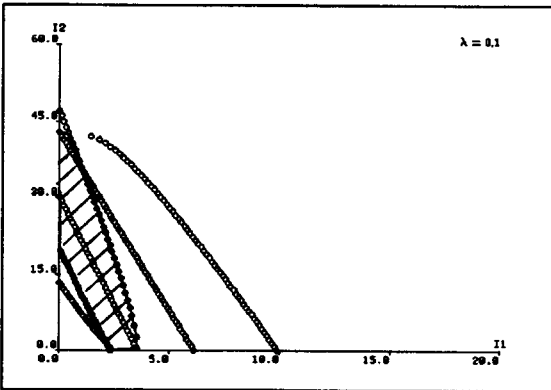
## Convex Hull



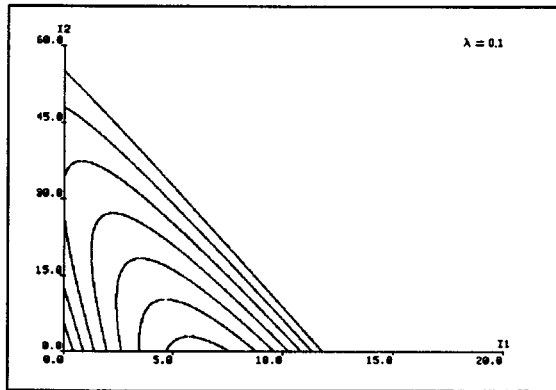
(a) First primary resonance zone  
The center line is the resonant actions.



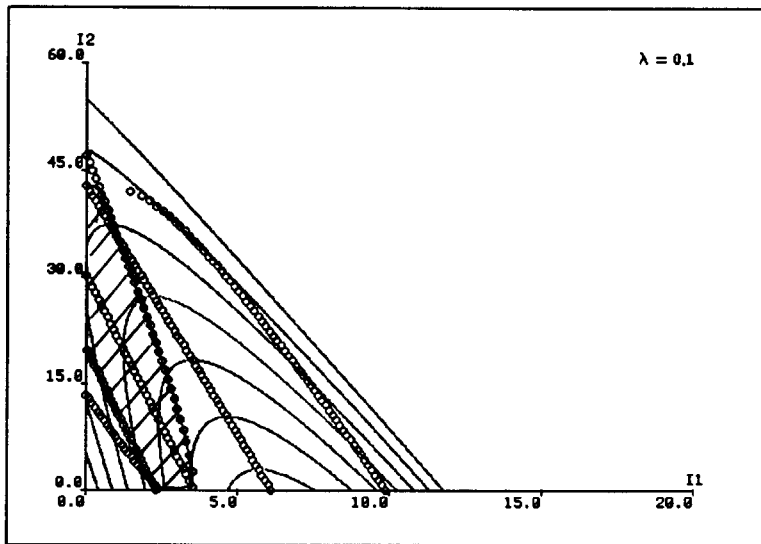
(b) Second primary resonance zone  
The center line is the resonant actions.



(c) Overlap region drawn as a convex polygon



(d) Level curves of  $H_0$



(e) Superposing (c) and (d) to obtain the resonance overlap diagram

Figure 6.12: Resonance overlap diagram for detuning frequency  $\lambda = 0.1$

Let  $S$  be the set of points that lie on the center line or the two boundaries of a resonance zone. The resonance zone is approximated by the **convex hull** of  $S$ , the smallest convex polygon containing all the points of  $S$ . The convex hull is computed by the **Graham scan** algorithm, which runs in  $O(n \log n)$  time where  $n$  is the number of points in  $S$  (see Shamos 1985). The output of the algorithm is the vertices of the convex hull arranged in the counterclockwise direction.

### **Intersection of convex polygons**

The resonance overlap region is approximated by the intersection of the convex hulls corresponding to the two resonance zones. Intersection of two convex polygons is computed by an algorithm invented by O'Rourke, which runs in  $O(n + m)$  time where  $n$  and  $m$  are the number of vertices of the two convex polygons [24]. The output of the algorithm is another convex polygon ordered in the counterclockwise direction.

### **Lines intersecting a convex polygon**

To compute the critical energy values at which large-scale chaos appear, KAM calculates which level curves of the hamiltonian intersect the overlap region. Because the level curves are unbounded – they are either parabolas or hyperbolas – any of them that intersects the overlap region will also intersect the boundary of the region. This observation makes the calculation of critical energy values easy. KAM walks along the convex polygon representing the overlap region. For each vertex of the polygon, KAM computes its corresponding energy. The critical energy values are the largest and smallest of all these energy values on the boundary.

### **Area of a polygon**

The size of the overlap region measures how chaotic the system is in the sense that the larger the overlap region is, the more initial conditions will lead to chaotic motion. The polygon representing the overlap region is divided into triangular pieces centered at its centroid. The area of the polygon is obtained by adding up the area of all these triangles. The area of a triangle is computed by Heron's formula, which says that the area of a triangle with sides of length  $a$ ,  $b$ , and  $c$  is equal to  $\sqrt{s(s-a)(s-b)(s-c)}$  where  $s$  is the semi-perimeter of the triangle.

## 6.6 COMPARING ANALYTICAL PREDICTIONS WITH NUMERICAL FINDINGS

Let's see how the resonance overlap diagram explains the banded and critical phenomenon observed in the numerical experiments.

If we put the level curves of the hamiltonian (Fig. 6.12c) and the overlap region (Fig. 6.12c) together, something interesting happens (Fig. 6.13). For small hamiltonian values (say,  $E < 2$ ), we see that the level curves do NOT intersect the resonance overlap zone. This suggests that isolated resonance zones dominate at low energies and so we should see only regular motion. When  $E$  is approximately 2.5, the level curves just touch the edges of the overlap zone, indicating the onset of chaotic motion. As  $E$  is increased, the level curves sweep across the interior of the overlap zone. We therefore expect widespread chaos in the Poincaré section for these values of the hamiltonian. For larger values of  $E$  (say,  $E > 5.9$ ), the level curves no longer intersect the overlap region. This predicts that regular motion becomes predominant again in the phase space in this regime of the hamiltonian. Comparing the predictions from the resonance overlap diagram with the numerical results in section 6.4, we find that the resonance overlap criterion gives results that are in fairly accurate agreement with those obtained from the numerical experiments. The resonance overlap diagram therefore explains the banded energy phenomenon.

Next we repeat our graphical analysis with the detuning frequency doubled (i.e.,  $\lambda = 0.2$ ). The results are summarized in Fig. 6.13. We have a similar resonance overlap region. But, unlike the previous case, the level curves are hyperbolas instead of parabolas. At low energy ( $E < 9$ ), the level curves are away from the overlap zone. We should therefore expect regular motion to be predominant. As  $E$  reaches a critical value, approximately 9.9, the level curves enter the overlap region. This suggests that the system becomes chaotic for large values of the Hamiltonian. Unlike the previous case ( $\lambda = 0.1$ ), the level curves never leave the overlap region once they are inside. This corresponds to the critical energy phenomenon that we have seen.

Despite the success of our the predictions, the resonance overlap criterion should not be expected to yield highly accurate results because of the approximations involved in deriving a simple expression for the resonance width. Nevertheless, because the method is general and the technique is easy to apply, it is a valuable tool for getting a "rough idea" of the dynamics of the phase space without detailed, time-consuming numerical simulations.

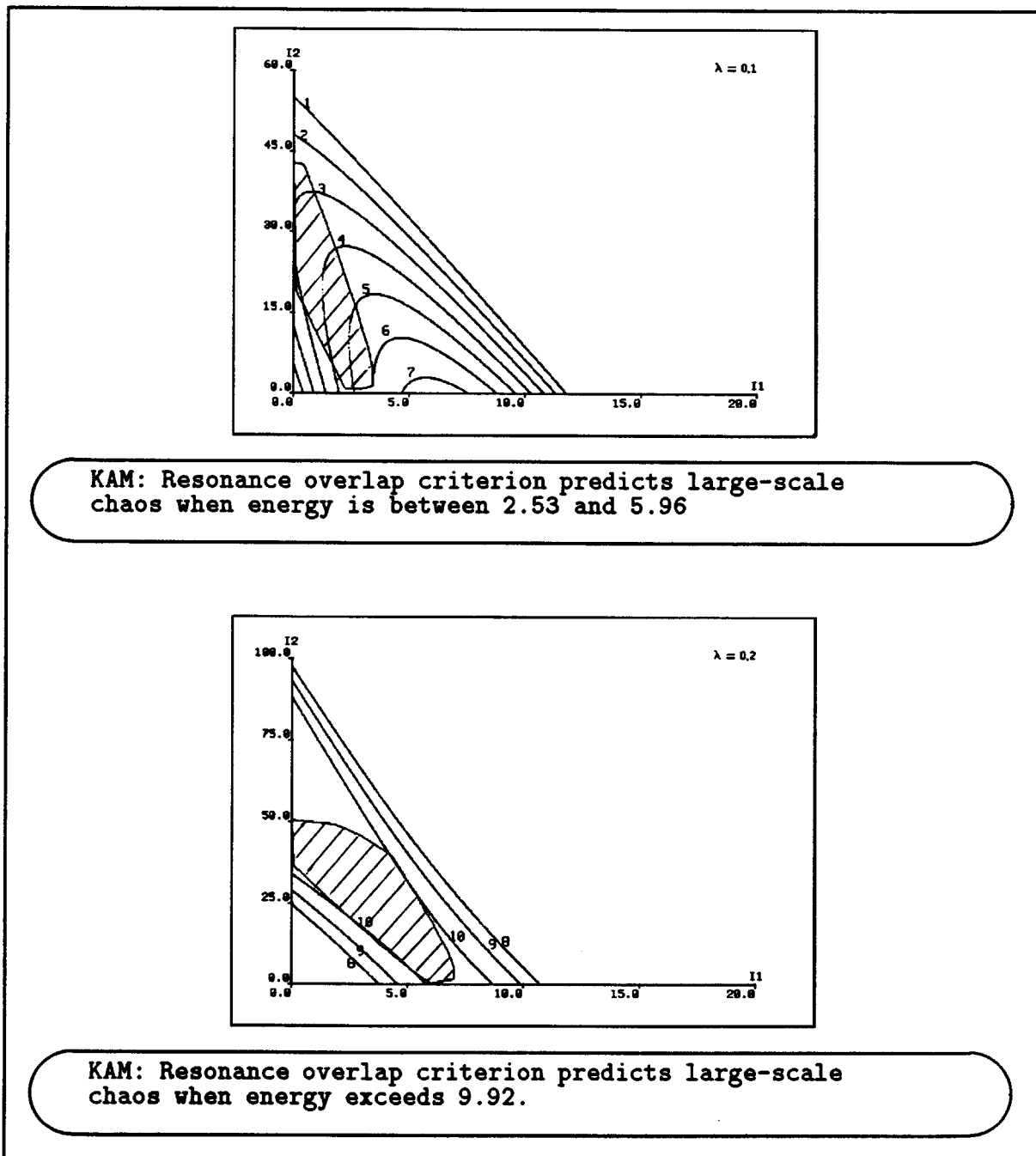


Figure 6.13: Predicting critical energy levels for large-scale chaos. Upper: The level curves of hamiltonian first touch the overlap region when  $E$  is between 2 and 3; they leave the overlap region when  $E$  is about 6. Lower: The level curves first touches the overlap region when  $E$  is about 10; they stay within the overlap region even when  $E$  is increased to the maximum allowable hamiltonian value.

## 6.7 HOW PHYSICAL PARAMETERS AFFECT CHAOS

### A Design Problem

Suppose we want to design a rectangular wave tank that exhibits chaotic motion under most initial conditions. Why might we want such a tank? Because chaotic motion allows rapid mixing of fluids, which may be quite useful for some purposes. For instance, we may desire rapid mixing of a dispersant with oil leaking from a ruptured tanker to prevent formation of large oil slick.

Solving the design problem requires some understanding of the effects of physical parameters, such as detuning frequency  $\lambda$ , driving amplitude  $\epsilon$ , water depth  $d$ , and length-width ratio  $l$ . In particular, we want to know what values of the physical parameters will lead to a maximum amount of chaos where the amount of chaos is measured by the fraction of the initial conditions in phase space that will lead to chaotic motion.

The design problem appears to be quite difficult. To begin with, the physical parameters are related in a very complicated way to the coefficients (such as  $\gamma_i, \Gamma_i, \Sigma$ ) in the amplitude equations. We do not have any analytic expressions, even approximate ones, that relate the physical parameters to the solution of the Hamiltonian problem. Short of help from analysis, we may resort to numerical simulations. But this is by and large infeasible too because of the high-dimensionality of the parameter space.

### Resonance overlap area as a measure of amount of chaos

The resonance overlap diagram suggests a new idea to search the parameter space. So far in analyzing the overlap diagram, we just use the fact that the level curves of the hamiltonian may or may not cross the overlap zone. But there is another piece of important information in the diagram – the *size* of the overlap zone. The size of the overlap zone measures the fraction of the initial conditions in the action space where chaotic motion dominates. It therefore looks like plotting the size of the overlap zone as a function of a physical parameter is going to be profitable. This idea is intuitively appealing because producing one overlap diagram takes about one minute of computer time whereas one picture of the Poincaré Map can take up to 12 hours on a Symbolics machine or a VAX 11/780.



Using the size of resonance overlap region as a measure for the extent of chaos, the following numerical experiments are performed. For each set of experiments, all but one physical parameter are fixed. KAM computes the overlap area for a few values of the parameters (the default is 8 parameter values). Then it interpolates the data points by a natural cubic spline to obtain a smooth curve. Figures 6.14 to 6.17 summarize the results of four sets of experiment.

The first experiment studies the effect of the detuning frequency  $\lambda$  on the size of the overlap zone. The range of  $\lambda$  is chosen to be between 0 and 0.4. The result is shown in Fig. 6.14. We see that the overlap area is small when the detuning frequency is small, but increases to a maximum around  $\lambda = 0.2$  and then falls off as  $\lambda$  is further increased. This is reasonable because  $\lambda = 0.2$  is approximately the resonant detuning frequency for the longitudinal wave.

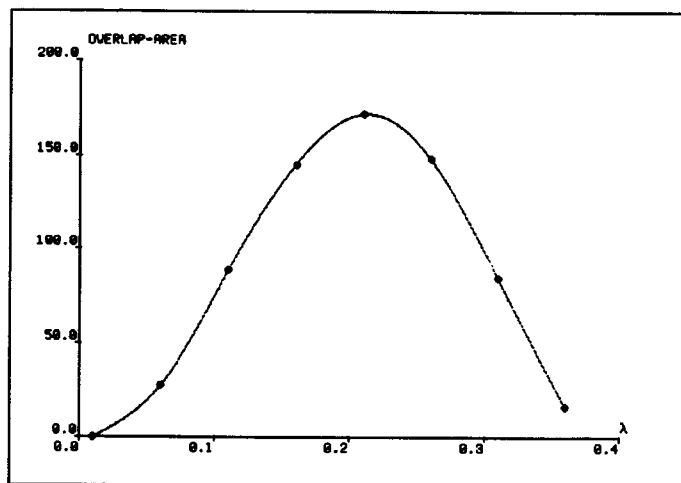


Figure 6.14: *Overlap area as a function of detuning frequency. The result is consistent with the linearized theory: a maximum response at the resonant frequency of the longitudinal wave.*

The second experiment studies the effect of the length-width ratio  $l$ . The result is presented in Fig. 6.15. The overlap area peaks at about  $l = 0.25$  and a much smaller peak is located at around  $l = 0.3$ . The prediction is consistent with the linear theory because  $l = 0.25$  is approximately the value at which the cross wave is excited.

The third experiment studies the effect of the water depth  $d$ . The result is displayed in Fig. 6.16. The overlap area attains a maximum when  $d$  is about 1.8. The figure also

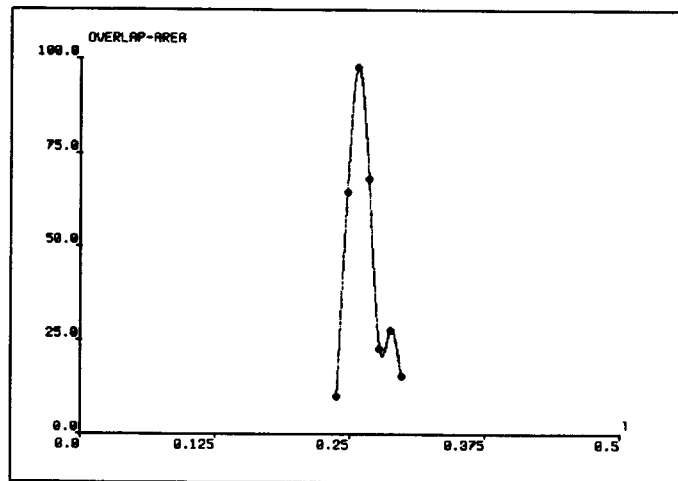


Figure 6.15: *Overlap area as a function of length-width ratio. The result is consistent with linearized theory: a maximum response at the resonant frequency of the cross wave.*

shows the overlap area slowly decreases as the water depth is increased. However, when the water depth is decreased, the overlap area rapidly shrinks and at about  $d = 1.36$ , the overlap area becomes zero. This is surprising for, unlike any of the previous cases, the overlap area vanishes not because the two resonance zones do not overlap, but because one of the resonance zones disappears (i.e., the resonance zone has zero width). It appears that a major bifurcation is occurring at the critical water depth  $d = 1.36$ .

The final experiment studies the effect of the driving amplitude  $\epsilon$ . The result is illustrated in Fig. 6.17. This time another unexpected result is obtained. For small driving amplitude, the overlap area is small. The area increases rapidly as  $\epsilon$  is increased until around  $\epsilon = 0.007$ . As  $\epsilon$  is further increased, the area actually *decreases*. This is counterintuitive; one might expect that the larger the driving amplitude, the larger the overlap area. At present, I do not have a good explanation for this phenomenon.

## 6.8 WHAT HAVE WE LEARNED?

Solving any realistic scientific problem requires knowledge, judgment, and attention to many details. To make rapid progress, a scientist must be able to focus on high-level, conceptual issues: decisions on what problems to attack, theory formulation, invention

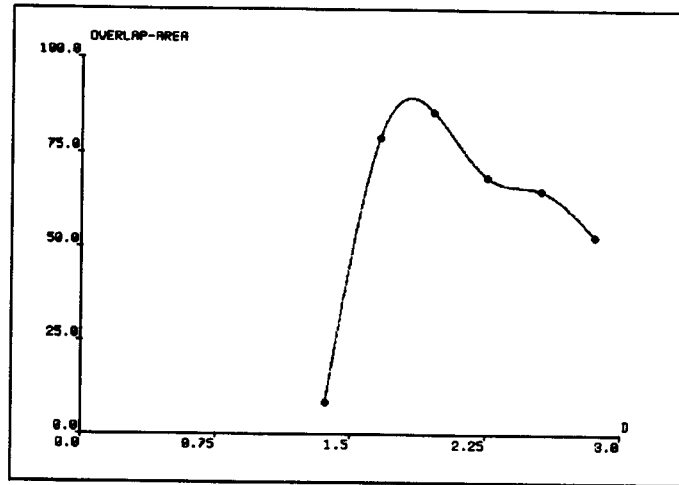


Figure 6.16: *Overlap area as a function of water depth.  $d$  (same as “ $D$ ” in the figure) = 1.36 is a critical water depth because one of the resonance zones completely disappears.*

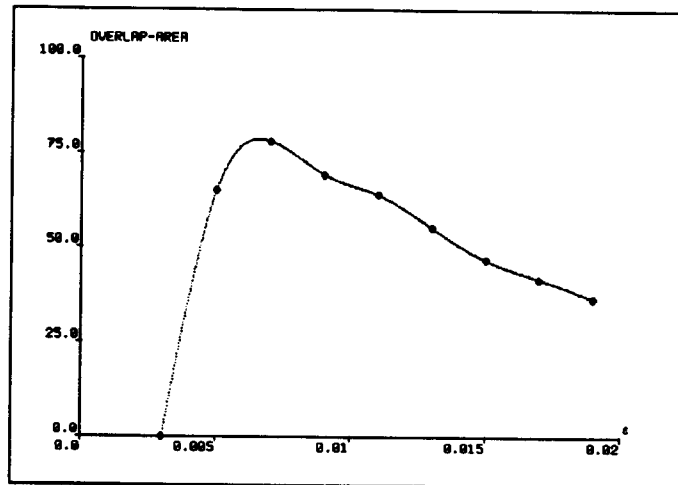


Figure 6.17: *Overlap area as a function of driving amplitude. The surprising result is that the extent of chaotic behavior decreases for larger driving amplitudes.*

of new techniques, and so on. It is therefore important to build computer programs like KAM that can act as useful assistants to the scientists.

What I have shown in this chapter is that it is already possible to automate many tasks

that used to occupy a significant portion of a scientist's (or his human assistants') time. Automatic planning and interpretation of numerical experiments, automatic application of approximation methods, automatic deduction of consequences from the approximation methods – these are the capabilities that KAM now has and that enable KAM to interact with a scientist in a high conceptual level.

The second conclusion is that building useful computer assistants for scientists is likely to be very difficult and time-consuming. Faster and bigger computers are, of course, helpful. But much more is needed. These computer programs must embody a large amount of knowledge about the physical domain and the underlying mathematics and geometric, symbolic, and numerical computation techniques.

## Chapter 6 Summary

This chapter presents a detailed case study: the analysis of standing waves in a wave tank. The case study serves two purposes: (1) demonstrating that KAM can solve problems that had not been considered when it was designed, and (2) showing how KAM might aid a scientist in the solution of an actual physical problem.

In addition to automatic phase space searching and parameter space searching, two new capabilities are incorporated into KAM:

- automatic compilation of a high-level description of the Poincaré section into numerical procedures that approximate the actions of the Poincaré map.
- automatic construction of a resonance overlap diagram, and deducing consequences from the diagram using geometrical algorithms.

With KAM's high-level assistance, I was able to complete the entire case study in a short time.

At present, KAM does not have any capability to assist a scientist in problem formulation. Nor is it able to decide which approximation methods to use.

# Chapter 7

## Conclusion

### 7.1 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

KAM can also apply an analytical approximation method, Chirikov's resonance overlap criterion, and generate resonance overlap diagrams, which are the basis for predicting the onset of chaos.

## 7.2 WHY DOES KAM WORK?

### 7.2.1 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 six possible orbit types: periodic orbit (including fixed point), quasiperiodic orbit, island chains, separatrix (or homoclinic orbit), chaotic orbit, and escape orbit.
- Each orbit type has a distinct geometry on the Poincaré section: a periodic orbit appears as isolated points; quasiperiodic orbit, a single closed curve; island chains, multiple isolated closed curves; separatrix, multiple connected loops; and chaotic orbit, a random splatter of points covering a region. Escape orbits will leave any bounded region in phase space.
- Orbits in phase space do not cross; consequently, on a 2-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.

## Knowledge of geometry

KAM knows some geometry – “know” in the sense of knowing how. This knowledge include 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.

### 7.2.2 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 algorithms, geometry knowledge in terms of production rules, and geometry knowledge in terms of algorithms – all these forms are stored in KAM. Fig. 7.1 shows examples of each type of representation.

	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 construct the resonance overlap diagram	Prim's algorithm to construct MST

Figure 7.1: *Examples of encoding knowledge in different forms.*



### 7.2.3 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 sequential 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.

## 7.3 WHAT ARE THE CONTRIBUTIONS?

### 7.3.1 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 the major contribution of this thesis.* These explicitly formulated constraints can be taught not only to a computer, but also to students of dynamics.

### 7.3.2 Computers, like people, need imagistic reasoning

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 non-verbal reasoning processes as part of the process for solving otherwise verbally presented problems. We observe scientists, engineers, and mathematicians continually using graphical representations to organize their thoughts about a problem. The programs I am developing use numerical methods as a means of shifting back and forth between geometric and symbolic methods of reasoning. KAM not only draws the orbits and phase portraits, but also looks at these pictures and holds them in its "mind's eye" so that powerful visual operations can be brought to bear on what otherwise would be purely symbolic problems.

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 [8] Geometry Theorem Proving Machine used diagrams to filter goals generated by backward chaining. Nevin's [21] forward-chaining theorem prover focused its forward deduction of facts on lines explicitly drawn in a diagram. Stallman and Sussman's [27] EL program performed antecedent deductions by exploiting the finite connectivity of devices. Finally, Novak's [22] ISAAC program used diagrams to solve word problems in physics.

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. 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 [18]. 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.

### 7.3.3 Intelligent systems are built by layers of specialized interpreters

The central organizing principle for KAM is that the designer must be free and completely flexible in specifying the processes that manipulate symbolic or geometric structures. Consequently, the system is built by separate layers of specialized interpreters. Each interpreter has its own representation language to encode knowledge, its own data

structure, and its own control. The interpreters communicate by passing interpreted symbolic objects. That these three interpreters turn out to have the same general structure – aggregation, partition, and classification – had not been anticipated when the system was designed. In general, the designer should be able to choose whatever programming technology seem appropriate, or even invent new ones, for the task at hand.

## 7.4 WHAT'S NEXT?

### 7.4.1 Enhancing KAM

KAM has several limitations:

- It treats the equations of motion as a black box; a smarter program would look at the form of the equations to see whether the equations have special properties, such as reversibility.
- It has a limited ability to describe the shape of a chaotic orbit; it would be nice if it can describe “holes” of a chaotic orbit independent of the underlying grid.
- It sets an arbitrary upper limit (1500) on the number of iterates that it is willing to look at; looking at more iterates will increase its orbit recognition ability.
- It cannot handle higher dimensional Hamiltonian systems for two reasons: first, new physics (such as Arnold diffusion) comes into play; second, the geometry of orbits become more complicated (e.g., a quasiperiodic orbit no longer partitions the phase space into two portions: an outside and an inside).
- It cannot handle multi-parameter Hamiltonian systems because we don't have a complete classification of bifurcations of periodic orbits in such systems.

### 7.4.2 Workbench Approach to Scientific Computing

KAM is just one of the many computational tools that scientists or engineers might want in solving their problems. For instance, they might want tools to help them formulate mathematical models, make analytical predictions, or obtain numerical solutions to more complicated mathematical problems.

For problem areas that are not well understood, they might want to use different tools at different stages of analysis. They might begin with a crude model that captures the most obvious effects, and then perform a few numerical experiments to gain a general understanding of the phenomenon. Armed with such understanding, they can often isolate the essential aspects of the phenomenon for analytical study. They might then compare the analytical predictions with more extensive numerical simulation and/or empirical data (see Fig. 7.2, an elaboration of part of Fig. 1.1 in Chapter 1).

The wave tank problem in Chapter 6 provided an example of such a modeling and analysis process. The complete mathematical model is given by the full Navier-Stokes equations with complicated boundary conditions. Unfortunately solving the full equations numerically is beyond the capabilities of the most advanced computers today. So he might make certain assumptions, like fluid incompressibility or zero dissipation, to obtain an approximate model. The approximate model, which is a simpler set of partial differential equations (PDEs), may still be too difficult or too expensive to solve numerically. For certain types of PDEs, he can apply analytical techniques to obtain the eigenvalue problems, which are a set of ordinary differential equations (ODEs). At this point, many analytical approximation methods and numerical methods can be used to solve the ODEs. In particular, smart ODE solvers like KAM can be used to completely automate the planning and interpretation of the numerical experiments.

My future goal is to produce a sophisticated scientist's workbench – a computer system that could automatically make approximation decisions, apply analytical techniques, plan and interpret numerical experiments, and compare analytical results with numerical and empirical data. No such workbench yet exists [1]. But for now I think it is our best hope for meeting the von Neumann Challenge.

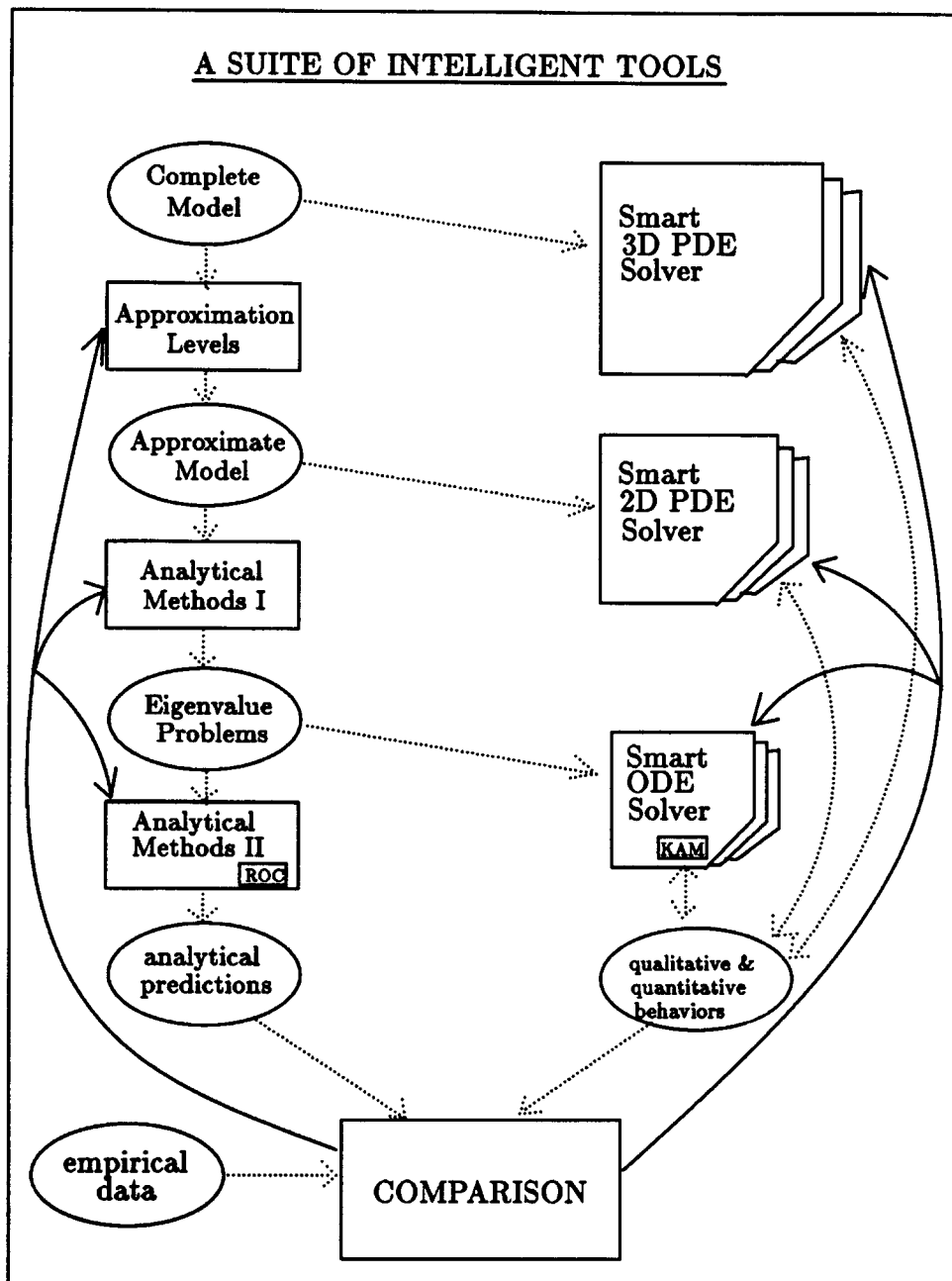


Figure 7.2: A picture showing how various intelligent tools (indicated by rectangles) might help a scientist in different stages of the modeling and analysis process. See also Fig. 1.1 of Chapter 1. KAM is just one of the many tools belonging to the category of smart ODE solvers. ROC stands for resonance overlap criterion, an analytical method discussed in Chapter 6. Dotted arrows indicate data flow; solid arrows, control flow.

# Appendix A

## User Manual

This appendix will explain what it is like to use KAM. KAM is a display-oriented system. You interact with KAM via a top-level window. I will first describe the structure of the display window, and then the basic capabilities of KAM.

### A.1 KAM WINDOW

Fig. A.1 shows the initial window when you first enter KAM. Now you can explore any area-preserving map by clicking on the top-level commands or by directly typing in these commands. The KAM window is divided into five panes: the picture pane, the action menu pane, the simulation statistics pane, the history of orbits pane, and the interaction pane.

#### Picture Pane

The picture pane is in the top-left corner of the window. Its purpose is presenting numerical results graphically. You can draw the axes of the phase space, display orbits, and show the current parameter value.

#### Action Menu Pane

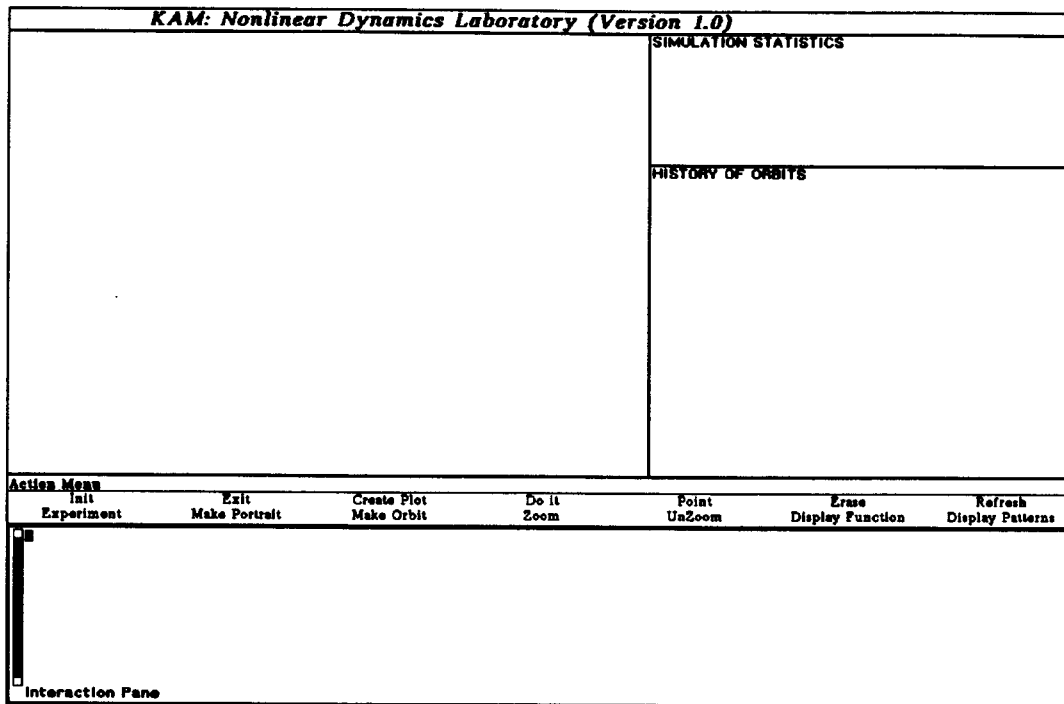


Figure A.1: *The initial window when you first enter KAM. The picture pane is the larger one on the upper left.*

The action menu pane is the horizontal pane in the middle of the window. It is a command processor pane. If you click on commands in the action menu, KAM will respond by first asking for the information needed to carry out the action, and then executing the action. You are not restricted to the commands in the menu. You can also type in commands or execute Lisp functions in the **interaction pane**.

### Simulation Statistics Pane

The simulation statistics pane is in the top-right corner of the window. It displays statistics – such as the number of orbits found, number of initial states tried, creation time of a phase portrait, and total simulation time used – related to phase space searching or parameter space searching

### History of Orbits Pane

The history of orbits pane is in the right-middle portion of the window. It is used in phase space searching. It shows a brief message about each of the orbits found so far during the phase space searching. The message includes three pieces of information: (1) a single character (“B” for bounding or “N” for non-bounding) indicating whether the orbit bounds an area on the Poincaré section or not, (2) type of the orbit (and a bound for its rotation number if the orbit is quasiperiodic), and (3) the time required to recognize it (Fig. A.8).

## Interaction Pane

The interaction pane is the horizontal pane in the bottom of the window. You can type in KAM commands or Lisp functions here. The pane is also used for displaying messages while KAM is running various tasks.

## A.2 TELLING KAM ABOUT THE MAPPING

Before running KAM, you need to tell it some information about the mapping that you are interested in exploring. To do this, you click on the INIT command in the action menu. The INIT command brings up a menu asking you for several pieces of information such as the name of the area-preserving map, the topology of the phase space, the names of the state variables, the range of state variables, and an escape range <sup>1</sup> (Fig. A.2). Another important piece of information is the energy surface constraint. The constraint specifies which part of the phase space is well-defined. We may need such a constraint if the mapping is not total, i.e., it is defined only on a subset of the phase space.

To create a correctly scaled plot of the phase space in the picture pane, you click on the CREATE PLOT command. The CREATE PLOT command asks you about the type of plot (e.g., whether you want a log-log plot or a linear plot), the labels on the axes, the starting values of the state variables on the axes, and so on (Fig. A.3). KAM will calculate the scaling factors based on the information about the size of the picture pane and the range of state variables.

---

<sup>1</sup>An escape range defines an escape region, which is given as the exterior of a closed and bounded set in the phase space. For instance, if you specify the closed interval  $[-2, 2]$  as the escape range, then the escape region is the exterior of the set  $S = \{ (x,y) : |x| \leq 2 \text{ and } |y| \leq 2 \}$



```

Choose Variable Values
Name of the iterate map: HENON-MAP69
Topology of phase space: EUCLIDEAN-PLANE
List of variables of the map: (X Y)
The parameter of the map: A
Range of value on the x-axis: (-1 1)
Range of value on the y-axis: (-1 1)
Escape range: (-2 2)
Energy surface constraint: NIL
Exit 

```

Figure A.2: Menu for the `init` command. In this particular example, the user is interested in exploring the mapping called *HENON-MAP69*. The phase space is the *EUCLIDEAN PLANE*, and he wants to show the portion of phase space inside the square  $[-1,1] \times [-1,1]$ . The escape range is given by  $[-2,2]$ . The mapping is defined on the entire phase space, so there is no energy constraint.

```

Choose Variable Values
Type of plot to be used: LOG-LOG-PLOT LOG-LIN-PLOT LIN-LOG-PLOT LIN-LIN-PLOT
Type of axes to be used: T-AXES PLUS-AXES L-AXES
Label of the x-axis: x
Label of the y-axis: y
Scale of the x-axis: 264.4
Scale of the y-axis: 184.8
Starting value of variable on the x-axis: -1
Starting value of variable on the y-axis: -1
Exit 

```

Figure A.3: Menu for the `create plot` command. The user specifies a linear plot. *KAM* automatically computes the scaling of the axes.

### A.3 FOUR MODES OF KAM

*KAM* can run in four different modes – mode 0 to 3. Mode 0 is the manual mode. In this mode, you have to make all the major decisions about finding orbits and phase portraits. Mode 3 is the completely automatic mode in which *KAM* automatically performs parameter space searching in a specified parameter range. Mode 1 and 2 are in-between: you and *KAM* share certain decision making in finding orbits and phase portraits. To run a particular mode, you click on the corresponding command in the **action menu** pane.

#### Mode 0: Iterating maps and displaying orbits

Mode 0 is the manual mode. In this mode, KAM graphically presents the result of iterating some mapping. To do this, you first click on POINT, which is displayed in the action menu. KAM will ask you to select an initial state in the phase space with the mouse cursor. After your selection, KAM displays the state you have chosen in the **interaction pane**. If the state is not what you want, then you can reselect another initial state by clicking on POINT again. Otherwise, you click on DO IT.

The DO IT command brings up a momentary menu for you to specify the values of several variables such as the number of iterates and the parameter value (Fig. A.4). You can modify the initial state by editing its value in the menu. The menu also asks you how often the orbit should be sampled. For example, you can tell KAM to keep every third iterate of the orbit. The default value is 1, i.e., all iterates are kept.

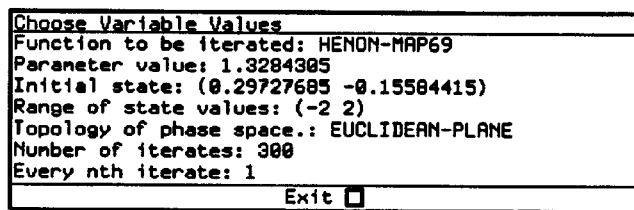


Figure A.4: Menu for the do it command. The user sets the parameter value to 1.3284305, and asks KAM to iterate the mapping 300 times.

By repeatedly using POINT and DO IT, you can manually search for interesting orbits in the phase space.

To zoom in a small region of the phase space, you can click on the ZOOM command. You will be asked to specify a rectangle in the phase space. KAM will magnify the rectangle in a **zoom pane**, which overlays the **picture pane**. The zoom-in facility is recursive. That means, you can specify another rectangle in the **zoom pane** for further magnification. The UNZOOM action undoes all the magnification, and returns to the original **picture pane**.

To erase the most recently drawn orbit, you can use the ERASE command.

### Mode 1: Automatic orbit recognition

In mode 1, KAM performs two tasks: (1) determines the number of the iterates that is needed for an orbit to reach steady state, and (2) identifies the type of the orbit. To

do automatic orbit recognition, you start by selecting an initial state using the POINT command. The initial state can also be specified explicitly by typing in the values of the state variables when you bring up the menu for the MAKE ORBIT command.

The MAKE ORBIT command asks you to specify the number of iterates to start (Fig. A.5). The default is 128. KAM will successively increase the number of iterates until a steady state is detected. When the steady state is reached, KAM identifies the type of the orbit.

```
Choose Variable Values
Initial state: (0.29727685 -0.15584415)
Initial number of iterates: 128
Parameter value: 1.3284305
Range of state values: (-2 2)
Topology of phase space.: EUCLIDEAN-PLANE
Increment in number of iterates: 128
Exit 
```

Figure A.5: Menu for the make orbit command. The user tells KAM to start with 128 iterates, and increment by 128.

Orbit recognition usually takes a few minutes. While KAM is engaged in orbit recognition, it will display various messages in the **interaction pane** to report its progress. The messages in Fig. A.6 are typical. Explanation of these messages will be given in Chapter 3.

## Mode 2: Automatic phase space searching

In mode 2, KAM automatically searches for representative orbits. In addition to recognizing orbit types, KAM decides what initial states to try. To start phase space searching, you click on the MAKE PORTRAIT command. The MAKE PORTRAIT command will first bring up a menu to ask you values of several variables (Fig. A.7). Most of these variables have already gotten their values from initialization or by default. You have to specify the parameter value for phase space searching, and an underlying grid (see section 4.5).

Phase space searching can take a few hours on a Symbolics . Many messages will be displayed in the **interaction pane** while the searching is going on. These messages inform you the current status of KAM. In the **history of orbits pane**, KAM shows the list of orbits found (Fig. A.8). When the orbits form a self-consistent phase portrait,

```

Making an orbit at (0.29727685 -0.15584415) indicated by an X.
Trying 128 iterates...
Inconsistent edges detected by Zahn's clustering = 0.
Total nodes = 128. Number of diameter nodes = 128. Branching nodes = 0
Diameter structure is (LINE).
Diameter length = 2.1644907. Diameter nodes = 128. Branching nodes = 0
Percentage of nodes on main stem and primary branches = 100.0%.
Percentage of branch nodes on diameter = 0.0%.
Clusters = 1. Inconsistent edges = 0. Longest inconsistent edge = 0.

Trying 256 iterates...
Inconsistent edges detected by Zahn's clustering = 0.
Total nodes = 256. Number of diameter nodes = 256. Branching nodes = 0
Diameter structure is (LINE).
Diameter length = 2.1759877. Diameter nodes = 256. Branching nodes = 0
Percentage of nodes on main stem and primary branches = 100.0%.
Percentage of branch nodes on diameter = 0.0%.
Clusters = 1. Inconsistent edges = 0. Longest inconsistent edge = 0.
STABILITY index is 100.0%.
The orbit reaches steady state at 256 iterates.
The iterates form a QUASIPERIODIC ORBIT.

```

Figure A.6: A typical progress report that KAM gives while it is performing automatic orbit recognition. After 256 iterates, KAM concludes that the orbit is quasiperiodic.

```

Choose Variable Values
Function to be iterated.: HENON-MAP69
Topology of phase space.: EUCLIDEAN-PLANE
Initial state: (0.29727685 -0.15584415)
Orbits to be added: NIL
Parameter value: 1.3284305
Energy surface constraint: NIL
Range of state values: (-2 2)
Number of grids on one side: 25
Range of x-values: (-1 1)
Range of y-values: (-1 1)
Display grids or not: Yes No
Exit 

```

Figure A.7: Menu for the make portrait command. The user tells KAM to explore the phase portrait with parameter value = 1.3284305. He does not give any initial orbits. So KAM starts with an empty phase portrait.

KAM terminates the search, and reports a summary describing the qualitative features of the phase portrait (Fig. A.9). Explanation of these messages will be given in Chapter 4.

**Mode 3: Automatic parameter space searching**

```

HISTORY OF ORBITS
[1] ==> (B) QUASIPERIODIC (5/26 1/5) 1.1 min. N= 256
[2] ==> (B) QUASIPERIODIC (4/19 3/14) 0.7 min. N= 256
[3] ==> (B) QUASIPERIODIC (5/24 4/19) 0.7 min. N= 256
[4] ==> (B) QUASIPERIODIC (1/5 7/94) 0.8 min. N= 256
[5] ==> (B) 5-ISLAND-CHAIN 1.4 min. N = 256
[6] ==> (B) 5-ISLAND-CHAIN 3.1 min. N = 304
[7] ==> (B) 5-loop SEPARATRIX 15.3 min. N = 000
[8] ==> (B) 5-ISLAND-CHAIN 3.3 min. N = 304
[9] ==> (B) ESCAPE [0] 0.9 min. N = 817
[10] ==> (B) QUASIPERIODIC (4/21 5/26) 1.3 min. N= 256

```

Figure A.8: KAM found 10 orbits. Each orbit is briefly described in the pane.

```

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.

```

Figure A.9: A typical summary report that KAM gives after it finds a self-consistent phase portrait.

Mode 3 is the completely automatic mode. KAM makes all the decisions – recognizing orbit types, searching representative orbits, and choosing parameter values – that are necessary for finding all the major bifurcations of fixed points and periodic orbits. To start parameter space searching, you click on the EXPERIMENT command. The EXPERIMENT command will bring up a menu asking you to specify values for several variables (Fig. A.10). Again most of these variables already have their default values. You only need to specify the parameter range for parameter space searching.

Parameter space searching usually takes a very long time. For example, finding all the major bifurcations for the Henon map in the parameter range (0, 2.2) takes about one

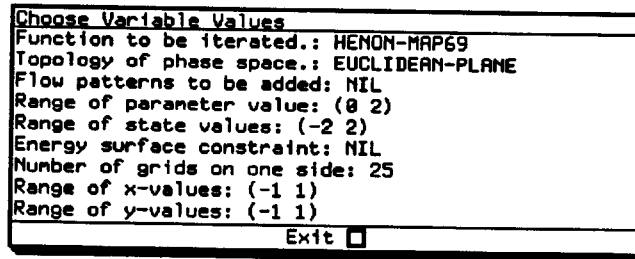


Figure A.10: Menu for the experiment command.

week of computer time. At the end of parameter space searching, KAM will summarize its findings in an executive report. Part of the executive report is about the major bifurcation patterns. A typical example of such report is shown in Fig. A.11.

In the parameter range (0 2.2), KAM finds a total of 38 phase portraits. The following bifurcation patterns are observed:

Poincare-Birkhoff bifurcation:  
 A pair of period-8 orbits appears at about 0.875 and the 8-island chain disappears at about 1.075.

Poincare-Birkhoff bifurcation:  
 A pair of period-7 orbits appears at about 1.000 and the 7-island chain disappears at about 1.157.

Poincare-Birkhoff bifurcation:  
 A pair of period-6 orbits appears at about 1.075 and the 6-island chain disappears at about 1.257.

Poincare-Birkhoff bifurcation:  
 A pair of period-5 orbits appears at about 1.271 and the 5-island chain disappears at about 1.437.

Poincare-Birkhoff bifurcation:  
 A pair of period-4 orbits appears at about 1.574 and the 4-island chain disappears at about 1.655.

Poincare-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.

Figure A.11: KAM generates a summary description of the major bifurcation patterns for the Henon Map.

# Bibliography

- [1] H. Abelson, M. Eisenberg, M. Halfant, M. Katzenelson, E. Sacks, G. Sussman, J. Wisdom, and K. Yip. Intelligence in scientific computing. *Communications of the ACM*, 32(5), 1989.
- [2] R Abraham and J. Marsden. *Foundations of Mechanics*. Benjamin-Cummings, 1978.
- [3] V.I. Arnold. *Mathematical Methods of Classical Mechanics*. Springer-Verlag, 1978.
- [4] Per Bak and M Hogh Jensen. Bifurcations and chaos in the  $\phi^4$  theory on a lattice. *Journal of Physics A*, 15, 1982.
- [5] Boris Chirikov. A universal instability of many dimensional oscillation systems. *Physics Reports*, 52, 1979.
- [6] Maxwell Clowes. On seeing things. *Artificial Intelligence*, 2, 1971.
- [7] Robert Devaney. *An Introduction to Chaotic Dynamical Systems*. Benjamin/Cummings Publishing Company, 1986.
- [8] H Gelernter. Realization of a geometry-theorem proving machine. In *Computers and Thought*. McGraw-Hill, 1963.
- [9] J Guckenheimer and P Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of vector fields*. Springer-Verlag, 1983.
- [10] G.H. Hardy and E.M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, fourth edition, 1960.
- [11] M. Henon. Numerical study of quadratic area-preserving mappings. *Quarterly of Applied Mathematics*, 27, 1969.
- [12] J Hochberg. *Perception*. Englewood Cliffs, second edition, 1985.

- [13] David Huffman. Impossible objects as nonsense sentences. In *Machine Intelligence 6*. Ellis Horwood Limited, 1971.
- [14] A.J. Lichtenberg and M.A. Lieberman. *Regular and Stochastic Motion*. Springer-Verlag, 1983.
- [15] J.D. Lin and L.N. Howard. Non-linear standing waves in rectangular tank due to forced oscillations. Hydrodynamics Laboratory Report 44, MIT, 1960.
- [16] Robert S MacKay. *Renormalization in Area Preserving Maps*. PhD thesis, Princeton University, 1982.
- [17] David Marr. *Vision*. W.H. Freeman and Company, 1982.
- [18] B. McCormick, T. Desanti, and M. Brown. Visualization in scientific computing. *Computer Graphics*, 21(6), 1987.
- [19] K.R. Meyer. Generic bifurcations of periodic points. *Transactions of American Mathematical Society*, 149, 1970.
- [20] John Miles. Parametrically excited, standing cross-waves. *Journal of Fluid Mechanics*, 188, 1988.
- [21] Arthur Nevins. Plane geometry theorem proving using forward chaining. *Artificial Intelligence*, 6, 1975.
- [22] G Novak. Representations of knowledge in a program for solving physics problems. In *Proceedings IJCAI-77*, 1977.
- [23] Seymour Papert. Theory of knowledge and complexity. In G.J. Dalenoort, editor, *Process Models for Psychology*. Rotterdam Univeristy Press, 1972.
- [24] Franco Preparata and Michael Shamos. *Computational Geometry*. Springer-Verlag, 1985.
- [25] R.C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 1957.
- [26] Russell Rimmer. *Generic bifurcations for involutory area preserving maps*. Memoirs of the American Mathematical Society, Number 272. American Mathematical Society, 1983.



- [27] Richard Stallman and Gerald Sussman. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9, 1977.
- [28] Ian. Stewart. *Does God Play Dice?* Basil Blackwell, 1989.
- [29] G.I. Taylor. An experimental study of standing waves. *Proceedings of the Royal Society*, 218A, 1953.
- [30] G.T. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12, 1980.
- [31] John von Neumann. On the principles of large scale computing machines. In *John von Neumann Collected Works Volume V*. MacMillan Co., 1963.
- [32] David Waltz. Understanding line drawings of scenes with shadows. In *The Psychology of Computer Vision*. McGraw-Hill, 1975.
- [33] C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 20, 1971.

*This blank page was inserted to preserve pagination.*

**CS-TR Scanning Project  
Document Control Form**

Date : 6/29/95

Report # AI-TR-1163

Each of the following should be identified by a checkmark:

Originating Department:

- Artificial Intelligence Laboratory (AI)
- Laboratory for Computer Science (LCS)

Document Type:

- Technical Report (TR)       Technical Memo (TM)
- Other: \_\_\_\_\_

**Document Information**

Number of pages: 235 (242-IMAGES)  
Not to include DOD forms, printer instructions, etc... original pages only.

Originals are:

- Single-sided or
- Double-sided

Intended to be printed as :

- Single-sided or
- Double-sided

Print type:

- Typewriter       Offset Press       Laser Print
- InkJet Printer       Unknown       Other: \_\_\_\_\_

Check each if included with document:

- DOD Form (2)       Funding Agent Form       Cover Page
- Spine       Printers Notes       Photo negatives
- Other: \_\_\_\_\_

Page Data:

Blank Pages (by page number): FOLLOWING PAGES #ED I, II

Photographs/Tonal Material (by page number): \_\_\_\_\_

Other (note description/page number):

Description :      Page Number:

- Ⓐ IMAGE MAG (1-10) PAGES I, UN#ED BLANK, II, UN#ED BLANK, III-VIII
- (11-235) PAGES #ED 1-225
- (236-239) SCANNED BY COLLECTOR, DOD (2)
- (240-242) TRGT'S (3)
- Ⓑ XEROX MARKS ON MOST PAGES

Scanning Agent Signoff:

Date Received: 6/29/95 Date Scanned: 7/10/95

Date Returned: 7/13/95

Scanning Agent Signature: Michael W. Cook

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AI-TR 1163	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) KAM: Automatic Planning and Interpretation of Numerical Experiments Using Geometrical Methods		5. TYPE OF REPORT & PERIOD COVERED technical report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Kenneth Man-kam Yip		8. CONTRACT OR GRANT NUMBER(s) N00014-86-K-0180 N00014-85-K-0124
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE August 1989
		13. NUMBER OF PAGES 225
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Distribution is unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES None		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) artificial intelligence                      scientific computing nonlinear dynamics                          scientific visualization numerical experiments                      imagistic reasoning		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) KAM is a computer program that can automatically plan, monitor, and interpret numerical experiments with Hamiltonian systems with two degrees of freedom. The program has recently helped solve an open problem in hydrodynamics - the prediction of onset of chaos in a resonantly excited rectangular wave tank of finite depth. Unlike other approaches to qualitative reasoning about physical system dynamics, KAM embodies a significant amount of knowledge about nonlinear dynamics. KAM's ability to control numerical experiments arises from (cont'd)		

## Block 20 cont's

the fact that it not only produces pictures for us to see, but also looks at (sic - in its mind's eye) the pictures it draws to guide its own actions. By combining techniques from computer vision with sophisticated dynamical invariants, KAM is able to exploit mathematical knowledge, represented in terms of a "grammar" that dictates consistency constraints on the structure of the phase space and parameter space. KAM is organized in three semantic levels: orbit recognition, phase space searching, and parameter space searching. 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.

# Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives**, using funds from the **Advanced Research Projects Agency** of the **United States Government** under Grant: **MDA972-92-J1029**.

The scanning agent for this project was the **Document Services** department of the **M.I.T. Libraries**. Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences**.

