INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI®

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

A COMPUTER ASSISTED APPROACH TO HILBERT'S 16TH PROBLEM

XIANHUA HUANG

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

MARCH 1999

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

0-612-39114-0

Canada

Dedicated to the memory of
my grandparents

Acknowledgements

I am indebted to my supervisor, Professor Eusebius. J. Doedel, for his great kindness and patience in guiding and helping my work of this thesis.

Abstract

A Computer Assisted Approach to Hilbert's 16th Problem

Xianhua Huang

A planar polynomial system of degree n is an autonomous system of ordinary differential equations

$$\begin{aligned} \dot{x} &= P_n(x,y), \\ \dot{y} &= Q_n(x,y), \end{aligned} \tag{1}$$

where $\dot{} \equiv d/dt$, and $P_n(x,y), Q_n(x,y)$ are relatively prime real polynomials, of degree at most n, one of which is of degree n. In the second part of the famous Hilbert 16th Problem, Hilbert asked for an upper bound $N(n)$ for the number of limit cycles of polynomial systems of degree n. Subsequent research has indicated that it is a remarkably intractable question.

In this thesis, we discuss a new approach to the Hilbert 16th problem via computer assisted analysis. In Chapter 1, we briefly recall the basic concepts of differential equations and the history of Hilbert's 16th problem. In Chapter 2, we describe multiparameter vectors, their bifurcations and rotated vector fields. In Chapter 3, we introduce parameter continuation methods and applications to multiparameter vectors. In Chapter 4, we summarize recent studies of quadratic systems and address the most used methods, including the uniqueness theorems and classifications of Hopf bifurcations. In Chapter 5, we mention examples of cubic systems having eleven limit cycles and the cubic systems of Liénard type. In Chapter 6, we apply parameter-continuation method to compute the limit cycle bifurcation diagram for quadratic systems of special interest, whose limit cycles can not be determined with the techniques of qualitative theory. Our computations support the assumption that quadratic systems have at most four limit cycles.In Chapter 7, parameter-continuation methods are applied to investigate some Liénard cubic systems. These systems are derived when codimension-three bifurcations are investigated and they are related to Hilbert's 16th problem. Bifurcation diagrams are obtained, in particular, some global bifurcations are seen for the first time. AUTO is used for bifurcation computations

and DSTOOL is used for phase portrait drawing. In addition, we have developed a Java program *QSYS*, which is used to visualize quadratic systems. In Appendix A, a brief user manual of *QSYS* is presented. In Appendix B, part of the Java source code of *QSYS* is given.

Contents

List of Figures

Chapter 1

Introduction

1.1 Hilbert's 16th problem

A polynomial system is a planar differential equations of the form

$$\begin{aligned} \dot{x} &= \sum_{k=0}^{n} \sum_{i+j=k} a_{ij} x^i y^j \equiv P(x,y), \\ \dot{y} &= \sum_{k=0}^{n} \sum_{i+j=k} b_{ij} x^i y^j \equiv Q(x,y). \end{aligned} \tag{2}$$

where $\dot{} \equiv d/dt$ and $a_{ij}, b_{ij} \in \mathbb{R}$, and $P(x,y)$ and $Q(x,y)$ are relatively prime real polynomials (i.e. they have no common factors), with at least one of them of degree n.

In the Second International Congress of Mathematicians, held in Paris in 1900, Hilbert listed 23 mathematical problems, which had a significant impact on twentieth century mathematics. In the famous list, Hilbert asked in the second part of the 16th problem for an upper bound on the number of limit cycles for nth degree polynomial systems, in terms of n (see Figure 1). The English translation reads as follows

16. Problem der Topologie algebraischer Kurven und Flächen.

Die Maximalzahl der geschlossenen und getrennt liegenden Züge, welche eine ebene algebraische Kurve $n$ter Ordnung haben kann, ist von Harnack[2]) bestimmt worden; es entsteht die weitere Frage nach der gegenseitigen Lage der Kurvenzüge in der Ebene. Was die Kurven 6ter Ordnung angeht, so habe ich mich — freilich auf einem recht umständlichen Wege — davon überzeugt, dafs die 11 Züge, die sie nach Harnack haben kann, keinesfalls sämtlich aufserhalb von einander verlaufen dürfen, sondern dafs ein Zug existieren mufs, in dessen Innerem ein Zug und in dessen Äufserem neun Züge verlaufen oder umgekehrt. *Eine gründliche Untersuchung der gegenseitigen Lage bei der Maximalzahl von getrennten Zügen scheint mir ebenso sehr von Interesse zu sein, wie die entsprechende Untersuchung über die Anzahl, Gestalt und Lage der Mäntel einer algebraischen Fläche im Raume* — ist doch bisher noch nicht einmal bekannt, wieviel Mäntel eine Fläche 4ter Ordnung des dreidimensionalen Raumes im Maximum wirklich besitzt.[3])

Im Anschlufs an dieses rein algebraische Problem möchte ich eine Frage aufwerfen, die sich, wie mir scheint, mittelst der nämlichen Methode der kontinuierlichen Koeffizientenänderung in Angriff nehmen läfst, und deren Beantwortung für die Topologie der durch Differentialgleichungen definierten Kurvenscharen von entsprechender Bedeutung ist — nämlich die Frage nach *der Maximalzahl und Lage der Poincaréschen Grenzcykeln (cycles limites) für eine Differentialgleichung erster Ordnung und ersten Grades* von der Form:

$$\frac{dy}{dx} = \frac{Y}{X},$$

wo X, Y ganze rationale Funktionen $n$ten Grades in x, y sind, oder in homogener Schreibweise

$$X\left(y\frac{dz}{dt} - z\frac{dy}{dt}\right) + Y\left(z\frac{dx}{dt} - x\frac{dz}{dt}\right) + Z\left(x\frac{dy}{dt} - y\frac{dx}{dt}\right) = 0,$$

wo X, Y, Z ganze rationale homogene Funktionen $n$ten Grades von x, y, z bedeuten und diese als Funktionen des Parameters t zu bestimmen sind.

Figure 1: The original statement of Hilbert's 16th problem, published in *Archiv der Mathematik und Physik (3) 1(1901) 213-237*

The maximum number of closed and separate branches which a plane algebraic curve of the nth order can have has been determined by Harnack. There arises the further question as to the relative position of the branches in the plane. As to curves of the 6th order, I have satisfied myself – by a complicated process, it is true – that of the eleven branches which they can have according to Harnack, by no means all can lie external to one another, but that one branch must exist in whose interior one branch and in whose exterior nine branches lie, or inversely. A thorough investigation of the relative position of the separate branches when their number is the maximum seems to me to be of very great interest, and not less so the corresponding investigation as to the number, form, and position of the sheets of an algebraic surface in space. Till now, indeed, it is not even known what is the maximum number of sheets which a surface of the 4th order in three dimensional space can really have.

In connection with this purely algebraic problem, I wish to bring forward a question which, it seems to me, may be attacked by the same method of continuous variation of coefficients, and whose answer is of the corresponding value for the topology of families of the curves defined by differential equations. This is the question as to the maximum number and position of Poincaré's boundary cycles (cycles limits) for a differential equation of the first order and degree of the form

$$\frac{dy}{dx} = \frac{Y}{X},$$

where X and Y are rational integral functions of the nth degree in x and y. ....

This problem remains unsolved, even for the simplest case, that of quadratic systems, i.e.

$$\begin{aligned} \dot{x} &= a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2 \equiv P(x,y), \\ \dot{y} &= b_{00} + b_{10}x + b_{01}y + b_{20}x^2 + b_{11}xy + b_{02}y^2 \equiv Q(x,y). \end{aligned} \tag{3}$$

where $P(x,y)$ and $Q(x,y)$ are relatively prime real polynomials, at least one of them of degree two.

Таким образом, мы получаем следующую теорему:
Теорема. *Уравнение*

$$\frac{dy}{dx} = \frac{P^*(x, y)}{Q^*(x, y)}, \qquad (1^*)$$

где P^* и Q^* — многочлены n-й степени, не может иметь более трех предельных циклов.

Figure 2: Landis & Petrovskii's result published in Matem. sbornik., Vol. 43, No. 2 (1957), 209-250

Extensive research on this problem started in the 1950's. The first approach was marked by the famous, unfortunately unreliable, paper by Petrovskii and Landis [41]. It claimed to have a solution to Hilbert's 16th problem for quadratic systems (See Figure 2). It reads: *We can conclude the following theorem: Differential equation*

$$\frac{dy}{dx} = \frac{P^*}{Q^*},$$

where P^* and Q^* are quadratic polynomials, has no more than three limit cycles. Later "refining" the analysis by Petrovskii and Landis, Cherkas insisted on the same conclusion([7]).

It seems not much attention was paid to the details of the proof by Petrovskii and Landis for quite some years. Not until the 1970's did some people review the article and found that some erroneous arguments were used. People tried to correct the argument, until counterexamples by Chen & Wang [6] (see 3) and Shi [53] (see 4) were given. According to the formula by Petrovskii and Landis, quadratic systems have at most three limit cycles. But Chen & Wang [6], and independently Shi [53], gave examples of quadratic systems which have at least four limit cycles. Thus the

research for a general solution to Hilbert's 16th problem was set back to the very beginning.

On the other hand, the qualitative study of concrete quadratic systems continued steadily. Assuming the work by Petrovskii and Landis, people further tried to give a partition of the coefficient space of quadratic systems in $\mathbb{R}^{12}$ and a complete description of the phase portraits for each coefficient region. Since the beginning of the century, and especially since the fifties, much research has been done on the problem of limit cycles in quadratic systems (see e.g. [6, 8, 9, 10, 25, 26, 33, 34, 48, 46, 47, 51, 52] and references therein). A complete bibiography was edited by J.W. Reyn [42]. Though the results were far from achieving the goal, even for quadratic systems, they provide important information. It was the research in this direction that led to the counterexample by Chen & Wang ([6]) and Shi ([53]), that shows that quadratic systems have at least four limit cycles. The research also indicated that the 16th problem is a remarkably intractable question. The evidence derived seems to suggest that quadratic systems have at most four limit cycles. Qualitative studies of cubic systems have also been carried out, but no significant results have been obtained yet toward Hilbert's 16th problem. It was found that there exist examples of cubic systems, which have eleven limit cycles. So far there are no examples of cubic systems with more limit cycles, but nobody knows whether the maximum number of limit cycles is indeed eleven. As for polynomial systems of higher degrees, very little work has been done. The most important results in this aspect are obtained by Lloyd [36], who studied small amplitude limit cycles. More precisely, he studied the number of weak foci and the number of limit cycles that can bifurcate from it, for polynomial systems of degree n.

In fact, it has only recently been firmly established that a given quadratic system has at most finitely many limit cycles [2]. The result for general polynomial systems, so-called Dulac Theorem, which was given by Dulac in 1950's with an incomplete proof, was finally proved a few years ago by Il'yashenko (See [28]). It states that any given polynomial system has finite number of limit cycles. However, Dulac's Theorem is a weakened assertion compared to Hilbert's 16th problem. Note that the number of limit cycles depends on the system. It does not imply the existence of a uniform upper bound for polynomial system of degree n. It is still unknown whether or not there exists a uniform upper bound for the number of limit cycles in the class of quadratic systems, not to mention the general Hilbert's 16th problem.

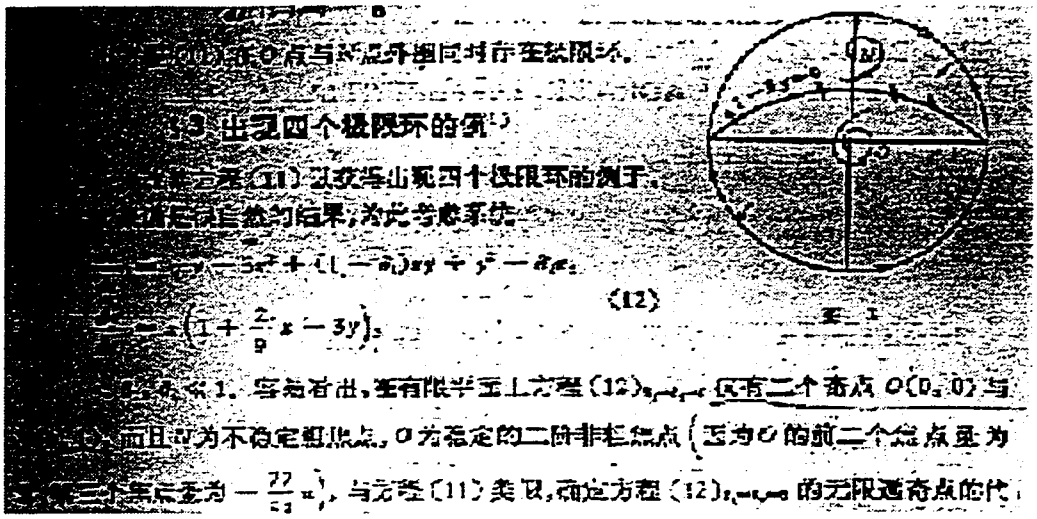

Figure 3: Chen & Wang's example of quadratic systems having (at least) 4 limit cycles, published in Acta Math. Sinica, Vol.22 (1979), 751-758.

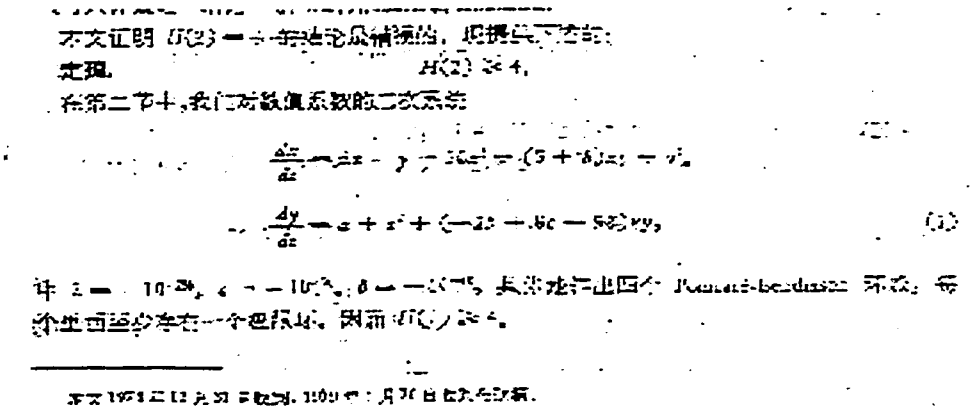

Figure 4: Shi's example of quadratic systems having (at least) 4 limit cycles, published in Scientia Sinica. (Ser. A) Vol.23, No. 2. (1980), 153-158

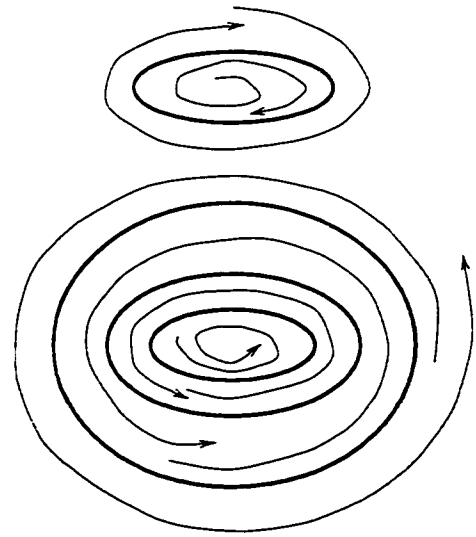

Figure 5: A four-limit-cycle configuration for a quadratic system

In the original statement of the 16th problem, the first part is about the number of algebraic curve branches, and the second part is about the number of limit cyles. It is natural to wish that limit cycle bifurcations would be algebraic, as Hilbert expected. But evidence from the Bogdanov-Takens bifurcation([4]) investigations suggests that the bifurcation of limit cycles can be non-analytic, even for quadratic systems. This brings a cloud for an algebraic approach.

Actually, there is no applicable method for estimating the number of limit cycles of polynomial systems. In particular, there is no method yet to prove that there are at most two or three limit cycles around a focus for a given system. The often used methods are those of classical analysis, with which various uniqueness results have been derived for quadratic systems. Mostly uniqueness theorems are formulated in terms of the famous Liénard equation. Fortunately, a quadratic system can always be brought into Liénard form and many interesting results about uniqueness of limit cycles have been obtained. For example, a quadratic system with an invariant straight line has at most one limit cycle (see [52]), and a quadratic system with a degenerate critical point has at most one limit cycle (see [10]).

The limitation of the methods forced people to seek weaker results.

Due to the difficulty to solve the Hilbert's 16th problem, Arnold posed the so-called weakened Hilbert 16th problem: How many reals zeros does the function $I(h)$

below have?

$$I(h) = \int\int_{H \le h} P(x,y)dxdy.$$

H is a real polynomial of degree n and P is a real polynomial of degree m. This problem is related to, but not equivalent to, the following limit cycle problem.

Let $H(x,y)$ be a polynomial of degree n, and let $F_1(x,y), F_2(x,y)$ be polynomials of degree $m+1$. What is the maximum number of limit cycles of the following system when ϵ is perturbed from zero?

$$\begin{aligned} \dot{x} &= \frac{\partial}{\partial y}H(x,y) + \epsilon F_1(x,y), \\ \dot{y} &= -\frac{\partial}{\partial x}H(x,y) + \epsilon F_2(x,y). \end{aligned} \tag{4}$$

The weakened Hilbert 16th problem is much simpler than the original 16th problem. Historically, it actually traces back to Poincaré. Poincaré studied the problem of a limit cycle created from a system with the origin O as a center. For this system, there is a family of closed orbits Γ_h covering a simply connected region in $\mathbb{R}^2$ including the origin O. The boundary of this region may be a singular closed orbit with critical points (including those at infinity). Each limit cycle of the perturbed system may take as its limit position either a closed orbit Γ_h of the unperturbed system, called *Poincaré bifurcation*; or the origin O of the unperturbed system, called *center bifurcation*; or a singular closed orbit of the unperturbed system, called *highly degenerate homoclinic or heteroclinic bifurcation.*

In particular, the weakened Hilbert 16th problem is conveniently dealt with by small parameter perturbation analysis. Still the problem is hard to solve. Very often the analysis leads to estimation of elliptic integrals, which can never be given explicitly.

1.2 Motivation

The traditional approaches to Hilbert's 16th problem can be classified into two categories: algebraic theory based on algebraic geometry and qualitative theory based on classical analysis. The algebraic theory approach can be traced back to the original statement of the 16th problem, but it achieved little, due to the non-algebraic nature of the problem as apparent from Bogdanov-Takens bifurcation [4]. The qualitative theory approach has achieved interesting results. Actually most of the results

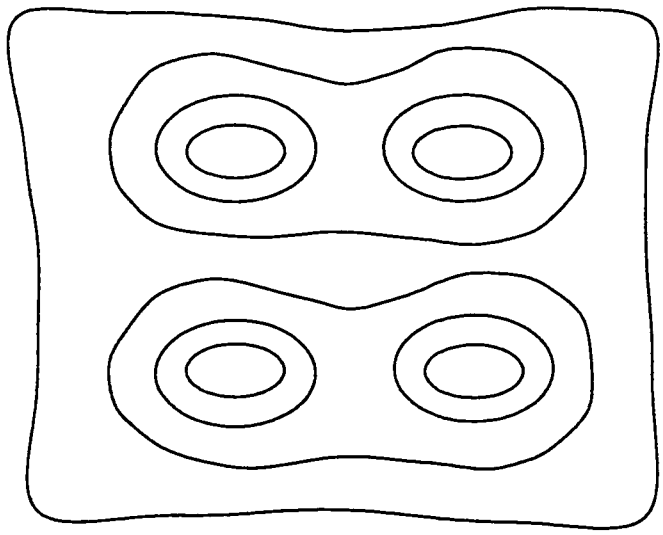

Figure 6: An eleven-limit-cycle configuration for a cubic system

about Hilbert's 16th problem are obtained by applying qualitative methods. But so far, qualitative theory cannot determine the number of limit cycles for parameters globally.

Given the difficulty to solve Hilbert's 16th problem, and the limitation of traditional methods, we here propose a new approach to the study of limit cycles when the number and ranges of the parameters are limited, say, for a system with 2 or 3 parameters which range in a finite interval. In the study of the 16th Problem, we sometimes arrive at a normal form for a given class of quadratic systems with 2 or 3 parameters (see reference [46, 49]). It can be shown that there exist limit cycles only for a small range of the parameters. But with qualitative theory, there is no way to determine the number of limit cycles. Thus we explore the use of computer assisted analysis to attack such problems. More precisely, we apply numerical methods that use a parameter continuation technique.

With the parameter continuation method, we are able to trace limit cycle bifurcations of one-parameter systems. We also can trace folds of double limit cycles for two-parameter systems. In fact, we can trace cusps of triple limit cycles for three-parameter systems. For quadratic systems, at most three limit cycles can bifurcate from a Hopf bifurcation. Thus it is interesting to see how the cusps, which corresponds to triple limit cycles, will evolve as the parameters change. The first goal of this approach is to visualize the bifurcation diagram of periodic solutions and understand the complexity. Further, we can do systematic computations for a normal form

of quadratic systems, and even for cubic systems.

1.3 Contributions

In this thesis, we obtain results that make new contributions to Hilbert's 16th problem. A concise summary is given below.

- We study quadratic systems with finite multiplicity two. It was conjectured that these quadratic systems have at most two limit cycles. Our analysis supports this conjecture. Moreover, we show a scenario: a semistable limit cycle originates from a second order weak focus and terminates at a homoclinic loop (see Section 1 of Chapter 6).

- We study quadratic systems with finite multiplicity four, in particular, the famous examples by Chen & Wang [6]. These examples demonstrate that quadratic systems can have at least four limit cycles. Our analysis supports that the Chen & Wang examples give at most four limit cycles. A semistable limit originates from the second order weak focus. It meets another limit cycle to become a limit cycle of multiplicity three as parameters change. It is no surprise that there are no more than three limit cycles (counting the multiplicities) surrounding a focus (see Section 2 of Chapter 6). The overall investigation provides more pieces of evidence to support that quadratic systems have at most four limit cycles.

- We study a cubic Liénard system in Chapter 7. Such systems are extracted from [15], where they are derived from the analysis of codimension three bifurcations. They are also related to Hilbert's 16th problem. The study of these bifurcations is partial. We obtain that in the focus/elliptic case a homoclinic loop can be surrounded by a limit cycle. We also obtain that in the saddle case there can be two limit cycles.

- We present a Java program QSYS, which is used to visualize quadratic systems (see Appendix A and B). It has a graphical user interface and allows easy viewing of the phase portraits of a quadratic system. It has the necessary functionalities: loading, saving, rescaling and printing. It is useful for people who are interested in Hilbert's 16th problem for quadratic systems.

1.4 Basic concepts

Consider a planar differential system

$$\begin{aligned} \dot{x} &= P(x,y), \\ \dot{y} &= Q(x,y), \end{aligned} \tag{5}$$

where $P(\cdot,\cdot), Q(\cdot,\cdot)$ are real differentiable functions on $\mathbb{R}^2$, and $\dot{} = d/dt$. The system is independent of time t, so it is an autonomous system. We always assume $P(\cdot,\cdot), Q(\cdot,\cdot) \in C^1$. The classical uniqueness theorem (see Theorem 1.1 in [57]) guarantees that the solutions of system (5) are unique with respect to a given initial value problem.

Definition 1.1 *If $(x,y) = (x_0,y_0)$ is a solution of the system of algebraic equations $P(x,y) = 0, Q(x,y) = 0$, then (x_0,y_0) is called a* **stationary solution** *of system (5).*

Definition 1.2 *Suppose $(x(t),y(t))$ is a solution of system (5). If $x(t), y(t)$ are periodic functions with the same period, say T, then we call $(x(t),y(t))$ a* **periodic solution** *with period T.*

Let $(x(t),y(t))$ be an arbitrary solution of system (5). Since the system (5) is autonomous, we can drawn this solution in $\mathbb{R}^2$ without self-intersection, unless it is a periodic solution. Moreover, all solutions of system (5) can be drawn in $\mathbb{R}^2$ without intersecting to each other. In particular, when being drawn in $\mathbb{R}^2$, a periodic solution of the system (5) is a simple closed curve.

Definition 1.3 *Suppose $(x(t),y(t))$ is a solution of system (5). When we draw it in $\mathbb{R}^2$, we say it is a* **orbit**. *When $(x(t),y(t))$ is a periodic solution, we get a simple closed curve geometrically. Such a geometrical configuration is called a* **closed orbit**. *When all the solutions of system (5) are drawn in $\mathbb{R}^2$, a closed orbit is said to be* **an isolated closed orbit**, *if there is no other closed orbit in its neighborhood. An isolated closed orbit, if exists, is called a* **limit cycle.**

Definition 1.4 *Suppose $(x_i(t),y_i(t)), i = 1,\cdots,k$ are solutions of system (5) for $t \in \mathbb{R}$ and suppose they are located in a finite part of the plane. If their end points match together and they form a simple closed curve, then we say that $(x_i(t),y_i(t)), i = 1,\cdots,k, t \in \mathbb{R}$* **form a singular closed orbit.**

Definition 1.5 *Let Γ be a limit cycle. We say Γ is* **exterior stable (inner stable)**. *if there is a neighborhood outside (inside) Γ, such that any orbit with initial position in the neighborhood approaches Γ as $t \to +\infty$. We say Γ is* **exterior unstable (inner unstable)**. *if there is a neighborhood outside (inside) Γ, such that any orbit with initial position in the neighborhood approaches Γ as $t \to -\infty$.*

The following theorem, the Poincaré-Bendixson Theorem, is well known (See Theorem 1.6 in [52])

Theorem 1.1 *Suppose Ω is a bounded closed domain in $\mathbb{R}^2$ and it contains a finite number of critical points of the system (5). If $\gamma_t^+ = \{(\phi_i(t), \psi_i(t)) : t \geq t_0\}$ is located in Ω, then its ω limit set $\gamma^+ \equiv \cap_{t>0}\gamma_t^+$ can be one of the following:*

- *a unique critical point;*
- *a unique closed orbit;*
- *a unique singular closed orbit.*

For example, system $dx/dt = -x - y, dy/dt = x - y$ has the origin $\{(0,0)\}$ as its ω limit set; system $dx/dt = x - y - x(x^2 + y^2), dy/dt = x + y - y(x^2 + y^2)$ has the unit circle $\{(x,y) : x^2 + y^2 = 1\}$ as its ω limit set.

Chapter 2

Multiparameter Vector Fields

The main purpose of this thesis is to study bifurcations of planar differential equations or vector fields with multiple parameters. So it is desirable to give an overview of what kind of bifurcations can happen. In this chapter, we will give a brief description of bifurcations for vector fields with multiple parameters. Emphasis will be put on bifurcations with multiplicity-two limit cycles and multiplicity-three limit cycles (Section 1) and those with a single parameter under so-called rotated vector fields (Section 2).

2.1 Multiparameter vector fields in the plane

Consider the following planar dynamical system with parameters $a \equiv (a_1, a_2, \cdots, a_n)$

$$\begin{aligned} \dot{x} &= X(x, y, a), \\ \dot{y} &= Y(x, y, a), \end{aligned} \tag{6}$$

where we assume that $X(\cdot,\cdot,\cdot), Y(\cdot,\cdot,\cdot)$ are analytic in the regions discussed below. This is the general form of a planar dynamical system. In particular, if we use system (6) to represent a polynomial system of degree m and a to represent the coefficients of the polynomial system, then $X(x, y, a), Y(x, y, a)$ are analytic in x, y, and even linear with respect to $a_1, a_2, \cdots, a_n$..

However, the analysis of limit cycles with multiple parameters is not a simple matter, even if we have very specific cases, such as quadratic systems or cubic systems.

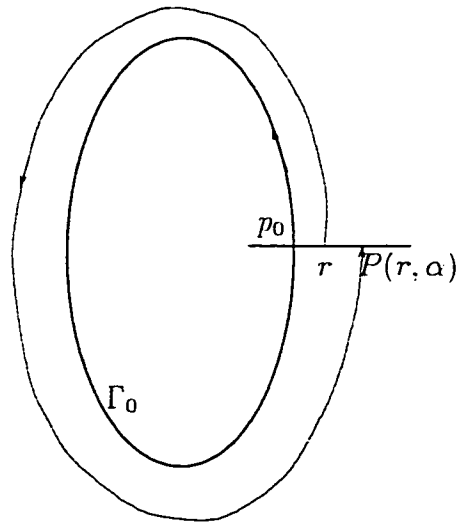

Figure 7: The Poincaré map $P(r,\alpha)$ and the displacement function $d(r,\alpha)$ for (6) and $||\alpha - \alpha_0|| \ll 1$

Assume that system (6) has a limit cycle Γ_0

$$\Gamma_0 : (x,y) = (\phi(t), \psi(t))$$

with period T_0 for $\alpha = \alpha_0 \in \mathbb{R}^n$. Let l be the straight line normal to Γ_0 at the point $p_0 = (\phi(0), \psi(0))$ and let r denote the coordinate along l with r positive on the exterior of Γ_0 and negative on the interior. By the continuity of of the initial value problem of differential equations, for fixed α_0 and small r, any orbit of system (6) starting from a point on l with distance r, will return to l, thus it defines a map $r \to P(r,\alpha_0)$ for sufficiently small r. By the continuity of solutions of system (6) with respect to parameters, the previous map can be extended for $||\alpha - \alpha_0|| << 1$. Moreover, because the system is analytical, $P(r,\alpha)$ should also be analytical. In other words, there exists a analytic function $P(r,\alpha)$ for $|r| << 1$ and $||\alpha - \alpha_0|| << 1$ with $P(0,\alpha_0) = 0$. The map $r \to P(r,\alpha)$ is called *Poincaré map*. The displacement function for system $(6)_\alpha$ along the normal line l to Γ_0 is then defined as the function

$$d(r,\alpha) = P(r,\alpha) - r.$$

(See Figure 7)

In terms of the displacement function, the limit cycle Γ_0 has multiplicity m iff

$$d(0,\alpha_0) = \frac{\partial d}{\partial r}(0,\alpha_0) = \cdots = \frac{\partial^{(m-1)} d}{\partial r^{(m-1)}}(0,\alpha_0) = 0.$$

and

$$\frac{\partial^{(m)} d}{\partial r^m}(0, \alpha_0) \neq 0.$$

By the qualitative theory of differential equations, the multiplicity m is independent of the choice of the point p_0 and the straight line l ([52]).

The following formulae, which determine the derivatives of the displacement function along Γ_0 are well-known ([52])

$$d_r(0, \alpha_0) = \epsilon^{\int_0^{T_0} div(X,Y)(\phi(t),\psi(t),\alpha_0)dt} - 1,$$

and

$$d_{\alpha_i}(0, \alpha_0) = c_{\pm} \int_0^{T_0} \epsilon^{\int_0^t div(X,Y)(\phi(t),\psi(t),\alpha_0)ds} \; (XY_{\alpha_i} - YX_{\alpha_i})(\phi(t), \psi(t), \alpha_0) \quad dt,$$

for $i = 1, 2, \cdots, n$, where

$$c_{\pm} = \frac{\pm 1}{|\sqrt{X^2 + Y^2}(\phi(0), \psi(0), \alpha_0)|}$$

and ± 1 is determined according to whether Γ_0 is clockwise or anti-clockwise.

Theorem 2.2 *Suppose that $n \geq 2$, and that for $\alpha = \alpha_0 \in \mathbb{R}^n$, the analytic system (6) has a multiplicity-two limit cycle Γ_0, and $d_{\alpha_1}(0, \alpha_0) \neq 0$. Then given $\epsilon > 0$, there is a $\delta > 0$ and a unique function $g(\alpha_2, \cdots, \alpha_n)$ with $g(\alpha_2^{(0)}, \cdots, \alpha_n^{(0)}) = \alpha_1^{(0)}$, well-defined and analytic for $|\alpha_2 - \alpha_2^{(0)}| < \delta, \cdots, |\alpha_n - \alpha_n^{(0)}| < \delta$.*

$$C_1 : \alpha_1 = g(\alpha_2, \cdots, \alpha_n)$$

is an $(n-1)$-dimensional, analytical bifurcation surface with a multiplicity two limit cycle passing through the point α_0 (See Figure 8).

Proof: Apply the implicit function theorem to the displacement function $d(r, \alpha)$. □

Theorem 2.3 *Suppose that $n \geq 3$, and that for $\alpha = \alpha^{(0)} \in \mathbb{R}^n$, the analytic system (6) has a multiplicity-three limit cycle Γ_0, and $d_{\alpha_1}(0, \alpha^{(0)}) \neq 0, d_{r\alpha_1}(0, \alpha^{(0)}) \neq 0$ and for $j = 2, \cdots, n$,*

$$\Delta_j \equiv \frac{\partial(d, d_r)}{\partial(\alpha_1, \alpha_j)}(0, \alpha_0) \neq 0.$$

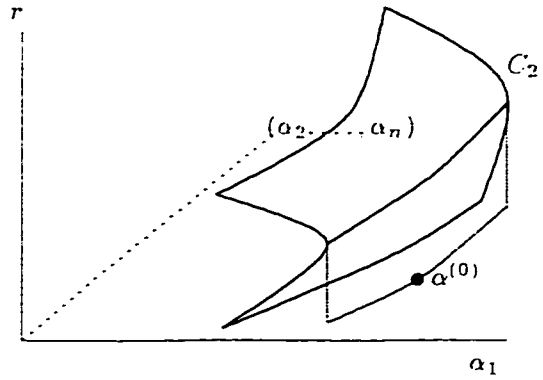

Figure 8: A bifurcation curve C_2 of multiplicity-two limit cycles of system (6)

Then given $\epsilon > 0$, there is a $\delta > 0$ and constant $\sigma_j = \pm 1$ for $j = 2, \cdots, n$, and there exist unique functions $h_1(\alpha_2, \cdots, \alpha_n), h_2(\alpha_2, \cdots, \alpha_n)$, $g_1(\alpha_2, \cdots, \alpha_n), g_2(\alpha_2, \cdots, \alpha_n)$ with $h_1(\alpha_2^{(0)}, \cdots, \alpha_n^{(0)}) = \alpha_1^{(0)}$, $h_2(\alpha_2^{(0)}, \cdots, \alpha_n^{(0)}) = \alpha_1^{(0)}$, $g_1(\alpha_2^{(0)}, \cdots, \alpha_n^{(0)}) = \alpha_1^{(0)}$, $g_2(\alpha_2^{(0)}, \cdots, \alpha_n^{(0)}) = \alpha_1^{(0)}$, where h_1, h_2 are well defined and analytic for $|\alpha_2 - \alpha_2^{(0)}| < \delta, \cdots, |\alpha_n - \alpha_n^{(0)}| < \delta$, and g_1, g_2 are well defined and continuous for $0 \le \sigma_j(\alpha_j - \alpha_j^{(0)}) < \delta$.

$$C_3 : \begin{cases} \alpha_1 = h_1(\alpha_2, \cdots, \alpha_n), \\ \alpha_2 = h_2(\alpha_2, \cdots, \alpha_n) \end{cases}$$

is an $(n-2)$-dimensional, analytical, cusp bifurcation surface with a multiplicity three limit cycle passing through the point α_0.

$$C_2^{1,2} : \alpha_1 = g_{1,2}(\alpha_2, \cdots, \alpha_n)$$

are two $(n-1)$-dimensional, analytical, fold bifurcation surfaces of multiplicity two limit cycles, which intersect in a cusp along C_3. (See Figure 9)

Proof: Apply the implicit function theorem (page 44 in [17]) for the existence of h_i and g_i (see Theorem 2 in [39]). □

2.2 Rotated vector fields

Now we consider the case of planar systems with one parameter. We rewrite the above system as follows

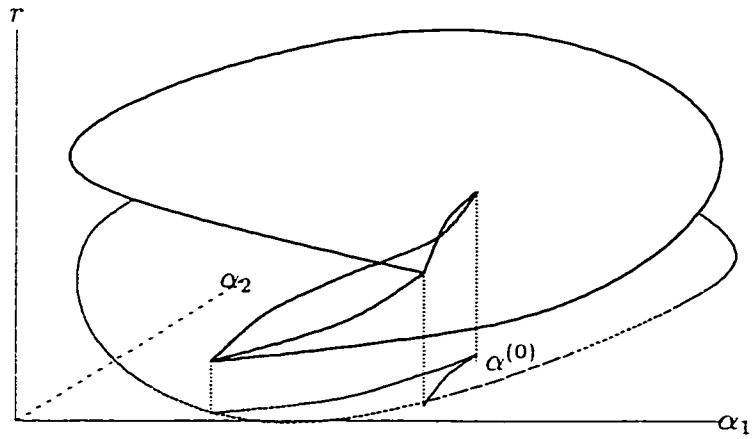

Figure 9: The bifurcation diagram of a multiplicity-three limit cycle for system (6) with $a \in \mathbb{R}^2$, which has a cusp bifurcation at $\alpha^{(0)}$

$$\begin{aligned} \dot{x} &= X(x, y, \lambda), \\ \dot{y} &= Y(x, y, \lambda), \end{aligned} \tag{7}$$

where $X(x, y, \lambda), Y(x, y, \lambda)$ are in $C^1(G \times I)$, G is a domain in $\mathbb{R}^2$, and I is an interval of $\mathbb{R}$.

Definition 2.6 *Suppose that*

i) for $\lambda \in I$, system (7) has its stationary solution fixed;

ii) for points $(x, y) \in G$ other than stationary solutions, and any $\lambda_1 < \lambda_2$ there holds

$$X(x, y, \lambda_2), Y(x, y, \lambda_1) - X(x, y, \lambda_1)Y(x, y, \lambda_2) \geq 0 (\leq 0).$$

but

$$\{(x, y) | X(x, y, \lambda_2), Y(x, y, \lambda_1) - X(x, y, \lambda_1)Y(x, y, \lambda_2) = 0 \text{ for any } \lambda_1, \lambda_2\}$$

does not contain any closed orbit (periodic solution) of (7),

then we say that system (7) forms a **generalized vector field.**

Here the condition (ii) can be strengthened as

$$X \partial Y / \partial \lambda - Y \partial X / \partial \lambda \geq 0 (\leq 0)$$

for those points that do not correspond to stationary and periodic solutions. The term *generalized vector field* is given to refer to the first definition of a rotated vector field defined by G.F. Duff, where more strict and slightly different conditions were required. A simple example of *generalized vector field* is as follows

$$\begin{aligned} \dot{x} &= f(x,y), \\ \dot{y} &= g(x,y) + \lambda f(x,y), \end{aligned} \tag{8}$$

where we assume that $f(\cdot,\cdot)$ is differentiable in $\mathbb{R}^2$ and $\{(x,y) : f(x,y) = 0\}$ does not contain any closed periodic solution of System (8).

In general, generalized vector fields have the following properties.

Theorem 2.4 *Assume that system (7) forms a generalized vector field with respect to the parameter λ. Then for different λ_1, λ_2, the periodic solutions of system $(7)_{\lambda_1}$ and $(7)_{\lambda_2}$ do not intersect.*

Proof: Assume that $\Gamma_i : x = \phi_i(t), y = \psi_i(t)$ is a periodic solution of system $(7)_{\lambda_i}$. Thus we have

$$\begin{aligned} \dot{\phi}_1(t) &= X(\phi_1(t), \psi_1(t), \lambda_1), \\ \dot{\psi}_1(t) &= Y(\phi_1(t), \psi_1(t), \lambda_1). \end{aligned} \tag{9}$$

By the definition of generalized vector field, we have that

$$\dot{\phi}_1(t)X(\phi_1(t), \psi_1(t), \lambda_2) - \dot{\psi}_1(t)Y(\phi_1(t), \psi_1(t), \lambda_2)$$

maintains a constant sign at the intersection points.

On the other hand, if the periodic solutions Γ_1 and Γ_2 are tangent at point $(\phi_1(t), \psi_1(t))$, then $\dot{\phi}_1(t)X(\phi_1(t), \psi_1(t), \lambda_2) - \dot{\psi}_1(t)Y(\phi_1(t), \psi_1(t), \lambda_2) = 0$. But this is impossible. If the periodic solutions Γ_1 and Γ_2 intersect, then there must be at least two intersection point: say, Γ_1 exits Γ_2 at one point, and enters at the other point. Thus $\dot{\phi}_1(t)X(\phi_1(t), \psi_1(t), \lambda_2) - \dot{\psi}_1(t)Y(\phi_1(t), \psi_1(t), \lambda_2)$ would have opposite sign at these points, which is a contradiction. □

Theorem 2.5 *Suppose System (7) forms a generalized vector field with respect to parameter λ, with for any $\lambda_2 > \lambda_1$,*

$$X(x,y,\lambda_2)Y(x,y,\lambda_1) - X(x,y,\lambda_1)Y(x,y,\lambda_2) \leq 0$$

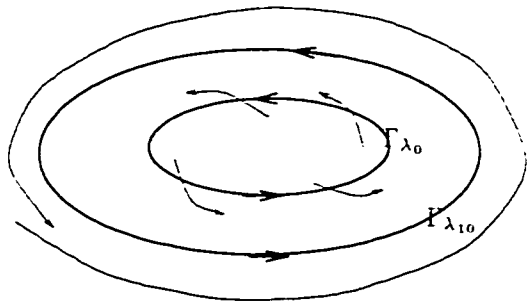

Figure 10: Periodic solution Γ_{λ_0} and perturbation under a rotated vector field: Γ_{λ_0}: anticlockwise and exterior stable periodic solution : $\Gamma_{\lambda_{10}}$: stable periodic solution, where $\lambda_{10} < \lambda_0$

in Condition 2. If for $\lambda = \lambda_0$ the system (7) has a periodic solution which is anti-clockwise (clockwise) and exterior stable, then for sufficient small $\epsilon > 0$, and there exists $\lambda_{10} < \lambda_0$ ($\lambda_{10} > \lambda_0$) such that in the $\epsilon-$neighborhood of Γ_{λ_0}, there exist an exterior-stable and an interior-stable periodic solution (both may coincide). Furthermore, there exists a $\delta-$neighborhood of Γ_{λ_0} (where $\delta \leq \epsilon$), filled with periodic solutions Γ_λ of system $(7)_\lambda$, and for $\lambda < \lambda_0$, no periodic solution of $(7)_\lambda$ exists inside Γ_{λ_0}. (See Figure 10)

Proof: The proof is similar to that of the previous theorem. □

Theorem 2.6 *Suppose system (7) forms a generalized vector field with respect to the parameter λ. If for $\lambda = \lambda_0$, the system (7) has a periodic solution which is anticlockwise and inner stable, then for sufficient small $\epsilon > 0$, there exists $\lambda_2 > \lambda_0$ such that in the $\epsilon-$neighborhood of Γ_{λ_0}, there exist an exterior-stable and an interior-stable periodic solution (which may coincide). Furthermore, there exists a $\delta-$neighborhood of Γ_{λ_0} (where $\delta \leq \epsilon$), filled with periodic solutions Γ_λ of system $(7)_\lambda$, and for $\lambda < \lambda_0$, no periodic solution of $(7)_\lambda$ exists outside Γ_{λ_0}.*

Proof: The proof is similar to that of the previous theorem. □

Theorem 2.7 *Suppose system (7) forms a generalized vector field with respect to parameter λ. Assume that $\lambda = \lambda_0$, the system has a semi-stable limit cycle Γ_{λ_0}, then when λ changes in one direction from λ_0, Γ_{λ_0} bifurcated into two periodic solutions:*

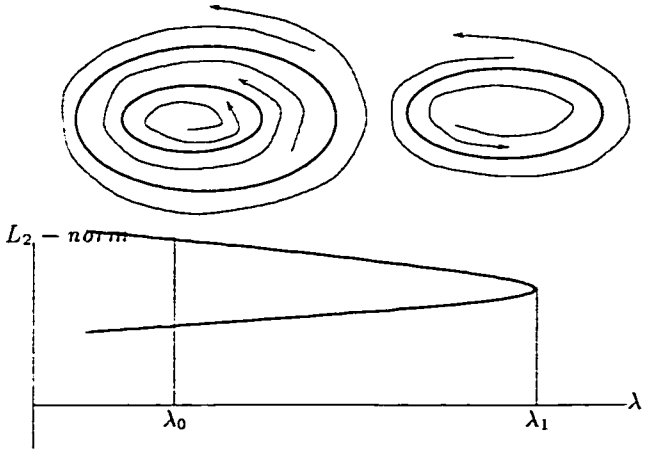

Figure 11: A semistable periodic solution bifurcation for a rotated vector field disappears as λ increases; and bifurcates into two periodic solutions as λ decreases

one is inside Γ_{λ_0}*; the other is outside* Γ_{λ_0}*. If* λ *changes in the other direction, from* λ_0*. the* Γ_{λ_0} *disappears.*

Proof: The proof is similar to that for the previous theorem. See [52] for details. □

Combining the above theorems, we can give the following description: Suppose system $(\bar{7})$ forms a generalized vector field with respect to parameter λ, and suppose that there are two adjacent periodic solutions: one is stable; the other one is unstable. Then, when λ increases or decreases, depending on the situation, the two periodic solutions will move closer and closer, and for some value of parameter λ, say $\lambda = \lambda_1$, they coincide and become a semi-stable periodic solution. When λ keeps changing from λ_1 in the same direction, then the semi-stable limit cycle disappears. (See Figure 11.)

The phenomenon described above can be viewed in another way: Draw a bifurcation diagram of periodic solutions, $L_2 - norm$ vs λ, as in Figure 11. For $\lambda = \lambda_0$, system $(\bar{7})$ has two periodic solutions: the outer is unstable and the inner is stable. The two periodic solutions exist for $\lambda \in (\lambda_0, \lambda_1)$. But at $\lambda = \lambda_1$, the two periodic solutions coincide and become one semistable periodic solution. After $\lambda > \lambda_1$, the system has no periodic solutions. If we construct a straight line segment which is transversal to the vector field, then we can define a return map of the vector field. The two periodic solution can be represented as two fixed points of the map. Thus we can describe the situation as follows: At $\lambda = \lambda_0$, the map has two fixed points.

which exist for $\lambda \in (\lambda_0, \lambda_1)$. When we trace one of them from $\lambda = \lambda_0$, the fixed points continue as λ increases. At $\lambda = \lambda_1$, there is a fold bifurcation, which corresponds to a semistable limit cycle of the underlying vector field. At this point, the fixed point turns back.

Often people have used rotated vector fields to describe the continuation of periodic solutions as a parameter changes. There are always conjectures that at certain parameter values, a system has a semistable periodic solution (limit cycle). But it is not possible to show the existence of a semistable limit cycle analytically. The difficulty is that a semistable periodic solution is a degenerate fixed point of the return map, which is difficult to determine by qualitative methods.

In the study of quadratic systems, it is interesting to know when a class of quadratic systems form a rotated vector field with a parameter. How will the periodic solution evolve as the parameter changes? In fact, there is no way to predict the behavior.

In the next chapter, we will introduce parameter continuation methods, which can be used to obtain semistable periodic solutions and which describe how the periodic solution changes as a parameter increases or decreases. We need parameter continuation for differential equations, but we can always discretize the differential equations and change them into algebraic equations. Therefore the following description will be given in terms of algebraic equations only.

Chapter 3

Parameter Continuation Methods of Periodic Solutions

The main contribution in this thesis is to apply parameter continuation techniques to the analysis of planar differential equations. In this chapter, we will give an introduction to parameter continuation methods. We summarize some algorithms for computing simple fixed points, fold points of algebraic equations and of periodic solutions of differential equations. We use AUTO for computing bifurcations.

Consider the following C^3 one-parameter equation

$$G(u, \lambda) = 0,$$

where

$$u, G(\cdot, \cdot) \in I\!R^n, \lambda \in I\!R.$$

Let $x \equiv (u, \lambda)$. Then the above equation can be written

$$G(x) = 0, \text{ with } G : I\!R^{n+1} \rightarrow I\!R^n.$$

Note that in the parameter formulation, $G(u, \lambda) = 0$, we have

$$\text{Rank}(G_x^0) = \text{Rank}(G_u^0 | G_\lambda^0) = n$$

if and only if

(i) G_u^0 is nonsingular, or

(ii) $\mathcal{N}(G_u^0) = 1$ and $G_\lambda^0 \notin \mathrm{R}(G_u^0)$.

Definition 3.7 *A solution x_0 of $G(x) = 0$ is* **regular**, *if $G_x^0 \equiv G_x(x_0)$ has maximal rank, i.e. if $Rank(G_x^0) = n$.*

A solution x_0 of $G(x) = 0$ is a **turning point** *or* **fold**, *if $\mathcal{N}(G_u^0) = 1$ and $G_\lambda^0 \notin R(G_u^0)$.*

Case i) can be dealt via a simple continuation method, which will be discussed in section 3.1; case ii), i.e. the case of a *turning point*, will be refined below.

For a regular solution, we choose ϕ such that $\mathcal{N}(G_x^0) = span\{\phi\}$; and ψ such that $\mathcal{N}((G_x^0)^*) = span\{\psi\}$.

Lemma 3.1 *Let $x_0 \equiv (u_0, \lambda_0)$ be a regular solution of $G(x) = 0$. Then, there exists a neighborhood U of the fold x_0 in $\mathbb{R}^{n+1}$, such that $G^{-1}(0) \cap U = \{(\lambda(s), u(s)) : |s - s_0| \leq \delta\}$, where $s \in \mathbb{R}, \delta > 0$, and $\lambda(\cdot), u(\cdot)$ are C^3-mappings satisfying $\lambda(s_0) = \lambda_0, u(s_0) = u_0$ and $|\lambda'(s)| + ||u'(s)|| > 0$ for every s and $\lambda'(s_0) = 0, u'(s_0) = \phi$.*

Definition 3.8 *A fold $(u_0, \lambda_0) \in \mathbb{R} \times \mathbb{R}^n$ is called* **a simple fold** *iff $\lambda''(s_0) \neq 0$; a fold $(u_0, \lambda_0) \in \mathbb{R} \times \mathbb{R}^n$ is called* **a double fold** *iff $\lambda''(s_0) = 0, \lambda'''(s_0) \neq 0$.*

Lemma 3.2 *1) A fold (u_0, λ_0) is simple iff*

$$a_2 = \psi^T(G_{uu}(u_0, \lambda_0)\phi\phi) \neq 0;$$

2) A fold (u_0, λ_0) is double iff

$$a_2 = \psi^T(G_{uu}(u_0, \lambda_0)\phi\phi) = 0$$

and

$$a_3 = \psi(G_{uuu}(u_0, \lambda_0)\phi\phi\phi + 3G_{uu}(u_0, \lambda_0)\phi) \neq 0.$$

where $r = u''(s_0)$ is the unique solution of

$$G(u_0, \lambda_0)r = -G_{uu}(u_0, \lambda_0)\phi\phi.$$

Proof: By direct calculation.□

Note: A cusp point is a double fold. But the converse is not true. We need an additional transversality condition, in order to define a cusp point as in catastrophe theory.([17, 19, 50])

3.1 Simple parameter continuation

To introduce parameter continuation methods, we start with algebraic equations. Suppose we have a solution (u_0, λ_0) of

$$G(u, \lambda) = 0.$$

as well as the direction vector $\dot{u}_0$. Here $\dot{u} \equiv du/d\lambda$. We want to compute the solution u_1 at $\lambda \equiv \lambda_0 + \Delta\lambda$.

Theorem 3.8 *Let $x_0 \equiv (u_0, \lambda_0)$ be a regular solution of $G(x) = 0$. Then, near x_0, there exists a unique one-dimensional continuum of solution $x(s)$ with $x(0) = x_0$.*

Proof: By implicit function theorem.□

To find the solution u_1, we use Newton's method

$$G_u(u_1^{(\mu)}, \lambda_1)\Delta u_1^{(\mu)} = -G(u_1^{(\mu)}, \lambda_1),$$
$$u_1^{(\mu+1)} = u_1^{(\mu)} + \Delta u_1^{(\mu)}.$$

where $\mu = 0, 1, 2, 3...$ Take $u_1^{(0)} = u_0 + \Delta\lambda\dot{u}_0$. If $G_u(u_1, \lambda_1)$ is nonsingular and $\Delta\lambda$ sufficiently small then the convergence theory for Newton's method assures us that this iteration will converge. After convergence, the new direction $\dot{u}_1$ can be obtained by solving

$$G_u(u_1, \lambda_1)\dot{u}_1 = -G_\lambda(u_1, \lambda_1).$$

This equation follows from differentiating $G(u(\lambda), \lambda) = 0$ with respect to λ at $\lambda = \lambda_1$.

The simple parameter continuation method can trace a fixed point only when the parameter monotonically changes. Thus, we can continue simple fixed points; but we cannot pass a simple fold. For differential equations, we can trace a periodic solution towards a semistable periodic solution, but we cannot pass the parameter value where the semistable periodic solution exists. Keller's method, which allows continuation through folds, will be discussed next.

Suppose we have a solution (u_0, λ_0) of $G(u, \lambda) = 0$, as well as the direction vector $(\dot{u}_0, \dot{\lambda}_0)$ of the solution branch. Keller's continuation consists of solving the following equations for (u_1, λ_1):

$$\begin{aligned} G(u_1, \lambda_1) &= 0. \\ (u_1 - u_0)^* \dot{u}_0 + (\lambda_1 - \lambda_0)\dot{\lambda}_0 - \Delta s &= 0. \end{aligned} \tag{10}$$

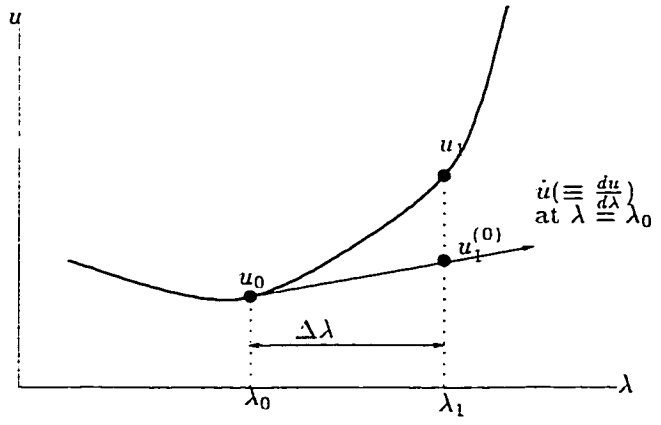

Figure 12: Graphical interpretation of Keller's continuation method

A graphical interpretation of Keller's method is given in Figure 12, where the first equation is the original equation; the second equation corresponds an approximation to the arc-length for parameter values between λ_0 and λ_1. To solve these equations, we use Newton's method:

$$\begin{pmatrix} (G_u^1)^{(\mu)} & (G_\lambda^1)^{(\mu)} \\ \dot{u}_0^* & \dot{\lambda}_0 \end{pmatrix} \begin{pmatrix} \Delta u_1^{(\mu)} \\ \Delta\lambda_1^{(\mu)} \end{pmatrix} = - \begin{pmatrix} G(u_1^{(\mu)}, \lambda_1^{(\mu)}) \\ (u_1^{(\mu)} - u_0) * \dot{u}_0 + (\lambda_1^{(\mu)} - \lambda_0)\dot{\lambda}_0 - \Delta s \end{pmatrix}.$$

The next direction vector can be computed from the following equations

$$\begin{pmatrix} G_u^1 & G_\lambda^1 \\ \dot{u}_0^* & \dot{\lambda}_0 \end{pmatrix} \begin{pmatrix} \dot{u}_1 \\ \dot{\lambda}_1 \end{pmatrix} = - \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

The above approach lets the computation continue with respect to pseudo-arclength. Thus, in principle, it allows the continuation pass a fold. Indeed, the above iteration scheme converges at simple folds. For details, see [11]. We can state

Theorem 3.9 *Let $x_0 \equiv (u_0, \lambda_0)$ be a regular solution or a simple fold of $G(x) = 0$. Then, near x_0, the computation can continue uniquely forward or backward, by using Keller's scheme.*

3.2 Computation of fold points

In this section, we discuss how to trace a curve of simple folds, starting from a simple fold solution (u_0, λ_0).

According to the earlier definition, a regular solution $x_0 \equiv (u_0, \lambda_0)$ of $G(u, \lambda) = 0$ is called a *simple fold* if

$$dim\mathcal{N}(G_u^0) = 1 \text{ and } G_\lambda^0 \notin \mathcal{R}(G_u^0).$$

Differentiating $G(u(s), \lambda(s)) = 0$, we have

$$G_u(u(s), \lambda(s))\dot{u}(s) + G_\lambda(u(s), \lambda(s))\dot{\lambda}(s).$$

In particular, we have

$$G_u^0 \dot{u}_0 = -\lambda_0 G_\lambda^0.$$

At a fold we have $G_\lambda^0 \notin \mathcal{R}(G_u^0)$. Thus we must have $\dot{\lambda}_0 = 0$. Hence $G_u^0 \dot{u}_0 = 0$, and since $dim\mathcal{N}(G_u^0) = 1$, we have

$$\mathcal{N}(G_u^0) = \text{ Span } \{\dot{u}_0\}.$$

Differentiating again, we have

$$G_u^0 \ddot{u}_0 + G_\lambda^0 \ddot{\lambda}_0 + G_{uu}^0 \dot{u}_0 \dot{u}_0 + 2G_{u\lambda}^0 \dot{u}_0 \dot{\lambda}_0 + G_{\lambda\lambda}^0 \dot{\lambda}_0 \dot{\lambda}_0 = 0.$$

At a simple fold (u_0, λ_0), let $\phi = \dot{u}_0$ then

$$\mathcal{N}(G_u^0) = \text{ Span } \{\phi\},$$

and also there exists a vector ψ such that

$$\mathcal{N}((G_u^0)^*) = \text{ Span } \{\psi\}.$$

Multiply by ψ^* and use $\dot{\lambda}_0 = 0$ and $\psi \perp \mathcal{R}(G_u^0)$ to find

$$\psi^* G_\lambda^0 \ddot{\lambda}_0 + \psi^* G_{uu}^0 \phi\phi = 0.$$

Here $\psi^* G_\lambda^0 \neq 0$ since $G_\lambda^0 \notin \mathcal{R}(G_u^0)$, thus we can solve

$$\ddot{\lambda}_0 = -\frac{\psi^* G_{uu}^0 \phi\phi}{\psi^* G_\lambda^0}.$$

To continue a fold, we need two free parameters. We use the extended system

$$\begin{aligned} G(u, \lambda, \mu) &= 0, \\ G_u(u, \lambda, \mu)\phi &= 0, \\ \phi^* \phi_0 - 1 &= 0. \end{aligned} \tag{11}$$

Here μ is a second parameter in the equations. The vector o_0 belongs to a "reference solution" $(u_0, o_0, \lambda_0, \mu_0)$. The above system has the form

$$F(U, \mu) = 0, \quad U \equiv (u, o, \lambda), \quad \text{with} \quad F : \mathbb{R}^{2n+1} \times \mathbb{R} \rightarrow \mathbb{R}^{2n+1}.$$

It can be shown that for fixed μ, $F_U(U, \mu)$ is a non singular matrix. Thus the Newton iteration technique applies to $F(U, \mu) = 0$. Therefore, we can conclude that

A simple quadratic fold can be continued in two parameters.

3.3 Computation of periodic solutions of differential equations

Next we introduce methods to compute periodic solutions of differential equations. Consider the following autonomous differential equation

$$\dot{u}(t) = f(u(t), \lambda) \tag{12}$$

where $u(\cdot), f(\cdot, \cdot) \in \mathbb{R}^n, \lambda \in \mathbb{R}$. We see a periodic solution of system (12) with period T. Fix the interval of periodicity by the transformation $t \rightarrow \frac{t}{T}$. Then the equation becomes

$$u'(t) = Tf(u(t), \lambda), \tag{13}$$

where $u(\cdot), f(\cdot) \in \mathbb{R}^n, T, \lambda \in \mathbb{R}$. Now seek solutions of period 1, i.e.

$$u(0) = u(1). \tag{14}$$

Assume that we have computed $(u_{k-1}(\cdot), T_{k-1}, \lambda_{k-1})$. We want to compute

$$(u_k(\cdot), T_k, \lambda_k) \equiv (u(\cdot), T, \lambda).$$

Equations (13) and (14) do not uniquely specify u and T, since $u(t)$ can be translated freely in time, i.e., if $u(t)$ is a periodic solution, $u(t + \sigma)$ is also a periodic solution. To specify a periodic solution for computation, we want the solution that minimizes

$$D(\sigma) \equiv \int_0^1 ||\tilde{u}(t + \sigma) - u_{k-1}(t)||_2^2 dt.$$

This optimal solution $\tilde{u}(t + \tilde{\sigma})$ satisfies the necessary condition $D'(\tilde{\sigma}) = 0$, i.e.

$$\int_0^1 (\tilde{u}(t + \tilde{\sigma}) - u_{k-1}(t)) * \dot{u}'(t + \tilde{\sigma}) dt = 0.$$

Writing $u(t) \equiv \tilde{u}(t + \tilde{\sigma})$ gives

$$\int_0^1 (u(t) - u_{k-1}(t)) * u'(t)dt = 0.$$

Integration by parts, using periodicity, gives

$$\int_0^1 u(t) * u'_{k-1}(t)dt \quad = \quad 0. \tag{15}$$

To trace out a branch of periodic solution, we use pseudo-arclength continuation. This allows calculation past folds along a branch of periodic solutions. Here the pseudo-arclength equation is

$$\int_0^1 (u(t) - u_{k-1}(t)) * u'_{k-1}(t)dt + (T - T_{k-1})\dot{T}_{k-1} + (\lambda - \lambda_{k-1})\dot{\lambda}_{k-1} = \Delta s. \tag{16}$$

Combining equations (13–16), we have a computational scheme that can be implemented. So we can conclude:

Suppose that we have computed a periodic solution of system (12) for some $\lambda = \lambda_0$. Then, combining equations (13–16), we can implement an algorithm of parameter continuation and trace out a branch of this periodic solution.

We have supposed that a periodic solution is given above. But this is not the case in general. Thus, to implement the above method in practice, we can let the computation start from a Hopf bifurcation.

Let (u_0, λ_0) be a Hopf bifurcation point of system (12). Assume that $f_u(u_0, \lambda_0)$ has a simple conjugate pair of purely imaginary eigenvalue $\pm i\omega_0, \omega_0 \neq 0$, and no other eigenvalues on the imaginary axis. Also assume that the conjugate pair of roots crosses the imaginary axis transversally with respect to λ. By the Hopf bifurcation Theorem, these conditions ensure the existence of a bifurcating branch of periodic solutions.

In the vicinity of a Hopf bifurcation point, the system (12) can be approximated by its linearization with constant coefficients

$$\phi'(t) \quad = \quad f_u(u_0, \lambda_0)\phi(t). \tag{17}$$

To compute a first periodic solution

$$(u_1, T_1, \lambda_1) \equiv (u, T, \lambda).$$

near a Hopf bifurcation (u_0, λ_0), we still have

$$u'(t) = Tf(u(t), \lambda_0). \tag{18}$$

$$u(0) = u(1). \tag{19}$$

Initial estimates for Newton's method are

$$u^{(0)}(t) = u_0 + \Delta s\phi(t), \quad T^{(0)} = T_0, \lambda^{(0)} = \lambda_0,$$

here $\phi(t)$ is nonzero solution of the time-scaled, linearized equations

$$\phi'(t) = T_0 f_u(u_0, \lambda_0)\phi(t), \quad \phi(0) = \phi(1),$$

that is,

$$\phi(t) = \sin(2\pi t)w_s + \cos(2\pi t)w_c,$$

where (w_s, w_c) is null vector in

$$\begin{pmatrix} -\omega_0 I, & f_u(u_0, \lambda_0) \\ f_u(u_0, \lambda_0), & \omega_0 I \end{pmatrix} \begin{pmatrix} w_s \\ w_c \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \omega_0 = \frac{2\pi}{T_0}.$$

The null space is generally two dimensional since $(-w_c, w_s)$ is also a solution. For the phase equation we align $u = u_1$ with $\phi(t)$, i.e.,

$$\int_0^1 u(t) * \phi'(t)dt = 0. \tag{20}$$

Since $\dot{\lambda}_0 = \dot{T}_0 = 0$, the pseudo-arclength equation for the first step reduces to

$$\int_0^1 (u(t) - u_0(t)) * \dot{\phi}(t)dt = \Delta s. \tag{21}$$

Now we can conclude that

The equations (18 - 21) can be used to compute a periodic solution of system (12) in the vicinity of a Hopf bifurcation point.

3.4 Implementation in AUTO

The computations of bifurcation diagrams in this thesis are done mainly by using software package AUTO. So it is desirable to know its interior algorithms. In this section, we will outline the the implementation in AUTO.

In last section, we have reduced ODE periodic solution problems to boundary value problem. Without loss of generality, we keep two parameters: λ, μ in the ordinary differential equations. Thus we have

$$u'(t) - f(u(t), \mu, \lambda) = 0, \ t \in [0, 1],$$

where

$$u(\cdot), f(\cdot) \in \mathbb{R}^n, \ \lambda \in \mathbb{R}, \ \mu \in \mathbb{R}^{n_1},$$

subject to boundary conditions

$$b(u(0), u(1), \mu, \lambda) = 0, \ b(\cdot) \in \mathbb{R}^{n_2},$$

and integral constraints

$$\int_0^1 q(u(s), \mu, \lambda) ds = 0, \ q(\cdot) \in \mathbb{R}^{n_3},$$

with

$$n_1 = n_2 + n_3 - n \geq 0.$$

We leave the parameter λ to be free for continuation. Introduce a mesh

$$\{0 = t_0 < t_1, \cdots < t_N = 1\}, \ \Delta_j \equiv t_j - t_{j-1}, \ (1 \leq j \leq N).$$

Define

$$\mathcal{P}_h^m = \{p_h \in C[0, 1] : p_h|_{[t_{j-1}, t_j]} \in \mathcal{P}^m\}.$$

where P^m is the space of polynomials of degree $\leq m$.

The collocation method consists of finding $p_h \in \mathcal{P}_h^m, \mu \in \mathbb{R}^{n_1}$ such that

$$p_h'(z_{j,i}) = f(p_h(z_{j,i}), \mu, \lambda)), \ j = 1, \cdots, N, \ i = 1, \cdots, m,$$

and such that p_h satisfies the boundary and integral conditions. In each subinterval $[t_{i-1}, t_j]$ the collocation points $z_{j,i}$ are the roots of the mth orthogonal polynomial.

Since each local polynomial is determined by $(m+1)n$ coefficients, the local number of degrees of freedom is $(m + 1)nN + n_1$. This is matched by the total number of equations:

- Collocation: mnN,
- Continuity: $(N - 1)n$,

- Constraints: $n_2 + n_3 (= n + n_1)$.

If the solution $u(t)$ is sufficiently smooth, then the global accuracy of this method is known to be of order m, i.e.,

$$||p_h - u||_\infty = \mathcal{O}(h^m).$$

At the main meshpoints t_j we have superconvergence:

$$max_j |p_h(t_j) - u(t_j)| = \mathcal{O}(h^{2m}).$$

The scalar variables μ are also superconvergent.

For each subinterval $[t_{j-1}, t_j]$ of the mesh, we introduce the Lagrange basis polynomials

$$\{\mathcal{L}_{j,i}(t)\}, \quad j = 1, \cdots, N, \quad i = 0, 1, \cdots, m,$$

defined by

$$\mathcal{L}_{j,i}(t) = \Pi_{k=0, k\neq i}^{m} \frac{t - t_{j-k/m}}{t_{j-i/m} - t_{j-k/m}}, \quad t_{j-i/m} \equiv t_j - \frac{i}{m}\Delta t_j.$$

The local polynomial can then be written

$$p_j(t) = \sum_{i=0}^{m} \mathcal{L}_{j,i}(t) u_{j-i/m},$$

and the collocation equations are

$$p_j'(z_{j,i}) = f(p_j(z_{j,i}), \mu, \lambda), \quad i = 1, \cdots, m, \quad j = 1, \cdots, N.$$

With the above choice of basis polynomials

$$u_j \approx u(t_j), \text{ and } u_{j-i/m} \approx u(t_{j-i/m})$$

where $u(t)$ is the solution of the continuous problem.

Based on the above, we can express the discrete boundary conditions, the integrals, and the pseudo-arclength equation in terms of the u-values.

The discrete boundary conditions are

$$b_i(u_0, u_N, \mu, \lambda) = 0, \quad i = 0, \cdots, n_2.$$

The integrals can be discretized as

$$\sum_{j=1}^{N}\sum_{i=0}^{m}\omega_{j,i}q_k(u_{j-i/m},\mu\lambda)=0,\quad k=1,\cdots,n_3,$$

where the $\omega_{j,i}$ are the Lagrange quadrature coefficients.

The pseudo-arclength equation is

$$\int_0^1(u(t)-u_0(t))^*\dot{u}_0(t)dt+(\mu-\mu_0)*\dot{\mu}_0+(\lambda-\lambda_0)\dot{\lambda}_0-\Delta s=0,$$

where

$$(u_0,\mu_0,\lambda_0),$$

is the previous computed point on the solution branch and

$$(\dot{u}_0,\dot{\mu}_0,\dot{\lambda}_0),$$

is the normalized direction of the branch at that point. The discretized pseudo-arclength equation is

$$\sum_{j=1}^{N}\sum_{i=0}^{m}\omega_{j,i}(u_{j-i/m}-(u_0)_{j-i/m})^*(\dot{u}_0)_{j-i/m}+(\mu-\mu_0)^*\dot{\mu}_0+(\lambda-\lambda_0)\dot{\lambda}_0)-\Delta s=0.$$

This completes the introduction to the implementation in AUTO.

Chapter 4

Limit Cycles of Quadratic systems

Most of the work in literature concerning Hilbert's 16th problem is on quadratic systems. In this chapter, we summarize some interesting results about quadratic systems (Section 1). These include that quadratic systems have at most two nests of limit cycles, that quadratic systems with two invariant straight lines have no limit cycles, and that quadratic systems may have four limit cycles with (1,3) distribution. We will mention the important role played by uniqueness theorems of limit cycles in the study of limit cycle and conjectures (Section 2). One important conjecture, which is related to uniqueness theorems is that if a quadratic system has two nests of limit cycles, then one of them contains at most one limit cycle. We also address the algebraic classification of quadratic systems (Section 3).

4.1 General properties of quadratic systems

Consider a quadratic system of differential equations in the plane

$$\begin{aligned} \dot{x} &= a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2 \equiv P_2(x,y), \\ \dot{y} &= b_{00} + b_{10}x + b_{01}y + b_{20}x^2 + b_{11}xy + b_{02}y^2 \equiv Q_2(x,y), \end{aligned} \tag{22}$$

where $a_{ij}, b_{ij} \in R$ and $P_2(x,y)$ and $Q_2(x,y)$ are relatively prime real polynomials of degree at most two (i.e. both $P_2(x,y)$ and $Q_2(x,y)$ have no common factors), at least one of them of degree two. Extensive research work on quadratic systems has been

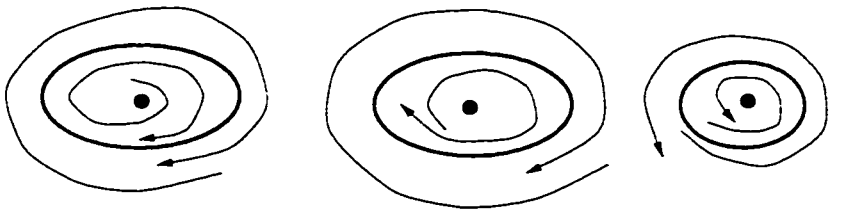

Figure 13: For a quadratic system, a limit cycle, if it exists, must contain a unique stationary solution. (a) an example of a stable limit cycle; (b) two nests of limit cycles

carried out since 1950's, in particular, by Chinese and Russian mathematicians. A complete bibliography was edited by J.W. Reyn ([42]).

Many interesting results have been obtained. We list a few of them below (see [52]). A straight line $l : ax + by + c = 0$ is called an invariant line of system 22, if any solution of system 22 being initiated at l remains on l.

Property 4.1 *If a quadratic system has two invariant straight lines, then it has no limit cycle.*

If a quadratic system has two invariant straight lines which are not parallel to each other, then we can choose them to be x-axis and y-axis, respectively. The system will then assume the following form

$$\begin{aligned} \dot{x} &= x(a_0 + a_1 x + a_2 y), \\ \dot{y} &= y(b_0 + b_1 x + b_2 y). \end{aligned} \tag{23}$$

This system is of Volterra type. Detailed analysis in [52] shows that such a system has no limit cycle.

Property 4.2 *A quadratic system has at most two nests of limit cycles, each of them surrounding at most one critical points. Furthermore, the stationary solution inside a limit cycle must be focus.*

A quadratic system has at most four stationary solutions in the finite part of the plane. When there exists a limit cycle, it must have a unique stationary solution inside (See Figure 13).

For a system of differential equations, a stationary solution undergoes a Hopf bifurcation when the divergence is zero at the stationary solution, while the determinant of the linearization is positive. In case of Hopf bifurcation, we can see either a family of closed orbits around the the stationary solution or a spiral around it. In the first case, we call the stationary solution a center; in the second case, we call such a Hopf bifurcation a weak focus. The foci can still be classified further, depending on how many limit cycles can bifurcate under generic perturbations. The maximum number of limit cycles generated is called the order of the focus.

Property 4.3 *A quadratic system can have weak foci up to order three. If a quadratic system has a weak focus of order three, it has at most one limit cycle, which surrounds some other non-weak focus.*

Property 4.3 follows from rather complicated analysis (See [52]). It implies that for quadratic systems, at most three limit cycles can bifurcate from a weak focus. It is expected that at most three limit cycles can surround a focus. Combining this with the fact that there are at most two nests of limit cycles, the maximum number of limit cycles should be six. But this is higher than what has been found. The following property follows from two famous papers ([6, 53]).

Property 4.4 *There exist examples of quadratic systems, where four limit cycles exist which are distributed in two nests: one contains one limit cycle; the other contains three limit cycles. In these examples, the quadratic systems have two real critical points and two complex conjugate critical points in the finite part of the plane.*

So far, the four limit cycles given in both, Chen & Wang [6] and Shi [53], is the maximum. No quadratic system with more than four limit cycles has been found. Moreover, evidence derived from qualitative analysis, or from the above properties, seems to suggest that there should be at most four limit cycles for any quadratic system.

Conjecture 1: *A quadratic system has at most four limit cycles.*

The above properties also suggest that if a quadratic system has two nests of limit cycles, then one contains one limit cycle, while the other one contains at most three.

Conjecture 2: *If a quadratic system has two nests of limit cycles, then one of them contains precisely one limit cycle.*

Conjecture 2 has been partially confirmed in [55], namely, for the case where the quadratic system has four critical points, which form a concave quadrangle ([55]), and that for the case of three critical points ([25]).

Furthermore, Conjecture 2 also implies that (1.3) is the only possible distribution of four limit cycles for quadratic systems and that there is no possibility for (2,2) distribution of limit cycle, provided that Conjecture 1 is true.

4.2 Uniqueness of limit cycles of quadratic systems

To study the limit cycles of quadratic systems, people usually transformed them into Liénard systems. Some very good uniqueness theorems have been formulated in terms of Liénard systems. These are applicable to quadratic systems and give very interesting results. Some are listed in the preceding section. To prove the uniqueness of a limit cycle around one critical point, we consider the following Liénard system

$$\begin{aligned} \dot{x} &= -g(y), \\ \dot{y} &= x - F(y), \end{aligned} \tag{24}$$

Uniqueness Theorem: *Suppose system (24) satisfies the following conditions:*

(i) $g(y) \in C^1$ *and* $F(y) \in C^2$ *for* $y \in (\alpha, \beta)$, *where* $\alpha < 0 < \beta$;

(ii) $yg(y) > 0$ *for* $y \in (\alpha, \beta)$ *and* $y \neq 0$;

(iii) there exists an α_1, *with* $\alpha < \alpha_1 < 0$, *such that* $f(y) > 0$ *for* $y \in (\alpha, \alpha_1)$ *and* $f(y) < 0$ *for* $y \in (\alpha_1, \beta)$, *where* $f(y) = F'(y)$;

(iv) $[f(y)/g(y)]' \geq 0$ *for* $y \in (\alpha, \alpha_1)$ *and* $y \in (0, \beta)$.

Then the system (24) has at most one limit cycle in the strip

$$D_1 := \{(x, y) | -\infty < x < \infty, \alpha < y < \beta\}.$$

If it exists, the limit cycle must be stable, and it is hyperbolic.

This is a typical uniqueness theorem for limit cycles, although some variations have been given and applied. In the application of such a uniqueness result, the use of classical analysis to verify the conditions is vital. Most of the analysis in the literature concerns this aspect.

There are many applications of uniqueness theorems such as the above, e.g., see [5, 8, 21, 25, 55]. Fortunately, there exists a general transformation to change a quadratic system into a Liénard system.

Consider the general quadratic system (22). Without loss of generality, we assume that P_2, Q_2 have no common factors; otherwise, system (22) becomes a linear system. By the theory of quadratic algebraic curves, there exists a real number λ such that ([16])

$$\lambda P_2(x, y) + Q_2(x, y) = 0$$

becomes a degenerate quadratic curve. Then by the transformation, $x_1 = x, y_1 = \lambda x + y$, we can change system (22) into

$$\begin{array}{lll} \dot{x}_1 & = P_2(x, y) & = P_2'(x_1, y_1) \\ \dot{y}_1 & = \lambda P_2(x, y) + Q_2(x, y) & = Q_2'(x_1, y_1) \end{array} \tag{25}$$

where $Q_2'(x, y) = 0$ either at one point in $\mathbb{R}^2$, or has no real solution, or can be factored as $R_1 R_2$, where R_1, R_2 are either linear or constant. For the first two cases, the system has no limit cycles by a general nonexistence theorem. Otherwise, we have that R_1, R_2 are linear and not constant simultaneously. Assume R_1 is not constant. Let $x_1 = R_1, y_1 = y + \lambda x$. Then we get the following system

$$\begin{array}{lll} \dot{x}_1 & = & P_2'(x_1, y_1) \\ \dot{y}_1 & = & x_1(ax_1 + by_1 + c). \end{array} \tag{26}$$

We can shift the coordinates so that $O(0, 0)$ is a critical point of the above system. This gives

$$\begin{array}{lll} \dot{x} & = & P_2(x, y) \\ \dot{y} & = & x(ax + by + c). \end{array} \tag{27}$$

where $P_2 = a_{10}x + a_{01}y + a_{20}x + a_{11}xy + a_{02}y^2$. This is the well known Ye form ([52]). We can conclude

Theorem 4.10 *All quadratic systems having limit cycles can be transformed into a system of Ye's form (27).*

The System (27) can always be transformed into a Liénard type system. Thus we can state

Theorem 4.11 *All quadratic systems having limit cycles can be transformed into a Liénard system.*

Proof: See [52].

Problems remain, even though we can transform any quadratic systems having limit cycles into a Liénard system, and apply its uniqueness theorems: The general transformation given in [52] results in a rather complex expression for $f(\cdot)$ and $g(\cdot)$. It is not easy to check the conditions in uniqueness theorems. In practice, we study the uniqueness problem for some quadratic systems with specific properties. It is always difficult to apply the uniqueness theorems, even though the methods used are classical.

4.3 Algebraic classification of quadratic systems

Next we consider the algebraic classification of quadratic systems in coefficient space. A critical point in the finite part of the plane is a common zero of $P(x, y)$ and $Q(x, y)$, which may be either real or complex. In general (22) has four finite critical points, real or complex, coinciding or not. In all these cases, the sum of the multiplicities of critical points, being the sum of the multiplicities of the common zeros of $P(x, y)$ and $Q(x, y)$, is less than or equal to 4. We will call this sum the *multiplicity* m_f of (22). Thus $m_f = 4$ in the generic case. By means of a linear transformation, thus yielding topologically equivalent phase portraits, the number of coefficients in (22) may be reduced. The class $m_f = 4$ still contains a considerable number of problems. By changing the coefficients appropriately, finite critical points can be sent to infinity which results in lowering of the finite multiplicity m_f of (22). The ensuring conditions lead to a further reduction of parameters. It seems natural for a systematic classification of phase portraits to start with the classes with lowest finite multiplicity and proceed to higher value of m_f. Such an approach enables us further isolate those problems still to be solved for 16th Hilbert's problem of quadratic systems.

In order to make further progress, one way to proceed is to give a classification of all phase portraits of quadratic systems, thereby going systematically through coefficient space to detect limit cycles. Another approach consists of studying the behavior of limit cycles in relation to certain properties of the systems, such as for example the presence of a center point, a degenerate point, a straight line solution, or other properties.

The presentation here involves both approaches. In classifying all possible phase portraits of quadratic systems, it seems natural to start, as is usually done in the analysis of a particular phase portrait, by investigating the possible number, location and character of the (finite and infinite) critical points of all systems, and ordering the various possible combinations of phase portraits and define classes of quadratic systems ([45]).

If the number of critical points of a quadratic system is finite, it has at most four finite critical points. Also, if some or all of them coincide or are complex, then the sum of their multiplicities is equal to four. In our terminology, we will call this sum the multiplicity m_f of a quadratic system. Thus $0 \leq m_f \leq 4$. For $m_f < 4$, one or more critical points "have gone to infinity", and since there can be only up to three locations of infinite critical points, these points will be of increasing complexity and multiplicity (up to 7) for small values of m_f. As a result, classes with a lower value of m_f can be described by a smaller number of parameters and contain a small number of phase portraits. In fact, the classification of phase portraits for $m_f = 0$ was given by Reyn [43], and these systems contain no limit cycles. That for $m_f = 1$ was given by the same author [44], and these systems contain at most one limit cycle. That for $m_f = 2$ was given by Reyn and Kooij [49]. In this case, the evidence is very strong that there can be at most two limit cycles. Articles [25, 46] dealt with the classification of phase portraits for $m_f = 3$, in particular, certain aspects of the limit cycle problem. Further evidence was provided that there are at most three limit cycles for a system in this class. It is interesting to mention that the examples of (at least) four limit cycles only appear in the class of quadratic systems with finite multiplicity four. This leads to a conjecture linking the maximum number of limit cycles for a quadratic system to its finite multiplicity.

Conjecture 3: *The maximum number of limit cycles for a given quadratic system equals its finite multiplicity* m_f.

Chapter 5

Limit Cycles of Cubic Systems

There are very few results about cubic systems concerning Hilbert's 16th problem. Efforts were made to find more limit cycles for cubic systems. Also the studies of codimension-three bifurcation lead to specific cubic Liénard systems [15]. In this chapter, two topics of cubic systems are addressed. The first topic is how eleven limit cycles arise in cubic systems (Section 1). The second topic is how qualitative theory is applied to specific cases to show the existence of at most two limit cycles, so called cubic Liénard systems, and that limitations are encountered for general cases (Section 2). These systems appear when codimension-three bifurcations are investigated ([15]).

The general cubic system is of the following form:

$$\begin{aligned} \dot{x} &= \sum_{k=0}^{3} \sum_{i+j=k} a_{ij} x^i y^j \equiv P(x,y), \\ \dot{y} &= \sum_{k=0}^{3} \sum_{i+j=k} b_{ij} x^i y^j \equiv Q(x,y), \end{aligned} \tag{28}$$

where "$\dot{}$" $\equiv d/dt$" and $a_{ij}, b_{ij} \in \mathbb{R}$, and $P(x,y)$ and $Q(x,y)$ are relatively prime real polynomials, at least one of them of degree three. The theory of algebraic curves implies the following

Property 5.1 *A cubic system has at most nine isolated critical points.*

Property 5.2 *If a cubic system has an ellipse as its invariant curve, then this system has at most one limit cycle.*

Property 5.3 *If a cubic system has five real invariant straight lines, then it can not have any limit cycle.*

It seems that the most interesting work for cubic systems is in two directions: one is to search for maximum number of limit cycles; the other is about limit cycles of specific cases, e.g., the Liénard system

$$\begin{aligned} \dot{x} &= y - F(x), \\ \dot{y} &= G(x), \end{aligned} \tag{29}$$

where $F(x)$ and $G(x)$ are real polynomials of x with degree three.

5.1 What is the maximum number of limit cycles for cubic systems

An interesting question is: what is the maximum number of limit cycles for cubic systems? Cubic systems are relatively complicated: as yet there is insufficient evidence for a conjecture. Some work has been done to explore how many limit cycles can be obtained for cubic systems. One result is given by Li & Huang [35], who found eleven limit cycles in a cubic system. This is the maximum number of limit cycles obtained for cubic systems up to now.

Li & Huang considered the following cubic system

$$\begin{aligned} \dot{x} &= y(1 - cy^2) + \mu x(mx^2 + ny^2 - \lambda), \\ \dot{y} &= -x(1 - ax^2) + \mu y(mx^2 + ny^2 - \lambda), \end{aligned} \tag{30}$$

where $a > c > 0, 0 < \mu << 1$, m and n are fixed, and λ is a free parameter. When $\mu = 0$, system (30) has nine finite critical points. They are

$$O(0,0), A_1^0(1/\sqrt{a}, 1/\sqrt{c}), A_2^0(-1/\sqrt{a}, 1/\sqrt{c}),$$

$$A_3^0(-1/\sqrt{a}, -1/\sqrt{c}), A_4^0(1/\sqrt{a}, -1/\sqrt{c}),$$

$$S_1^0(0, 1/\sqrt{c}), S_2^0(0, -1/\sqrt{c}),$$

$$S_3^0(-1/\sqrt{a}, 0), S_3^0(1/\sqrt{a}, 0),$$

where $O, A_1^0, A_2^0, A_3^0, A_4^0$ are centers; and $S_1^0, S_2^0, S_3^0, S_4^0$ are saddle points. When $0 < \mu << 1$, the above nine critical points persist: we denote them as O, A_i, S_i $(i = 1, 2, 3, 4)$. It is easy to see all these critical points are located on the curve

$$ax^4 + cy^4 - (x^2 + y^2) = 0.$$

To determine limit cycles, we consider the perturbing system (30) from $\mu = 0$. We want to determine where Poincaré bifurcation exists, i.e., how many of the family of periodic solutions persist after μ changes from 0. It has been shown that, when $a = 2, c = 1, m = 1, n = -3, \lambda = .5, \mu = .003$, by perturbing system (30) from $\mu = 0$, we can get cubic systems which have 11 limit cycles (See Figure 14). The phase diagram in Figure 14 is not clear enough. Actually, we can get eleven limit cycles through careful analysis. There are two limit cycles around A_1^0, A_2^0, A_3^0, and A_4^0, respectively. There is one limit cycle around group A_1^0, A_2^0 and S_1^0, one around the group A_3^0, A_4^0 and S_2^0. And there is a large limit cycle aroud the group of all nine critical points: $O, A_1^0, A_2^0, A_3^0, A_4^0, S_1^0, S_2^0, S_3^0$ and S_4^0

Open Problem: Can we get more than eleven limit cycles for Cubic systems?

5.2 Limit cycles of cubic Liénard systems

The Liénard system (29) appears when codimension-three bifurcations are investigated ([15]). System (29) has three critical points whose $x-$coordinates are given by the solution of $G(x) = 0$. They can be either three real solutions (counting multiplicity) or one real solution and two conjugate complex solutions. If $x = x_0$ is a solution of $G(x) = 0$, then $(x, y) = (x_0, F(x_0))$ is a critical solution of the Liénard system (29). To investigate limit cycles of the Liénard system (29), we distinguish the cases where a limit cycle contains either one critical point inside or more. For the case that a limit cycle contains a unique critical point, we are only able to determine the uniqueness of the limit cycle in some situations using, using the uniqueness theorem stated earlier in this chapter.

For cubic Liénard systems, there are new tools to prove the existence of at most two limit cycles. The following theorem to determine at most two limit cycles looks

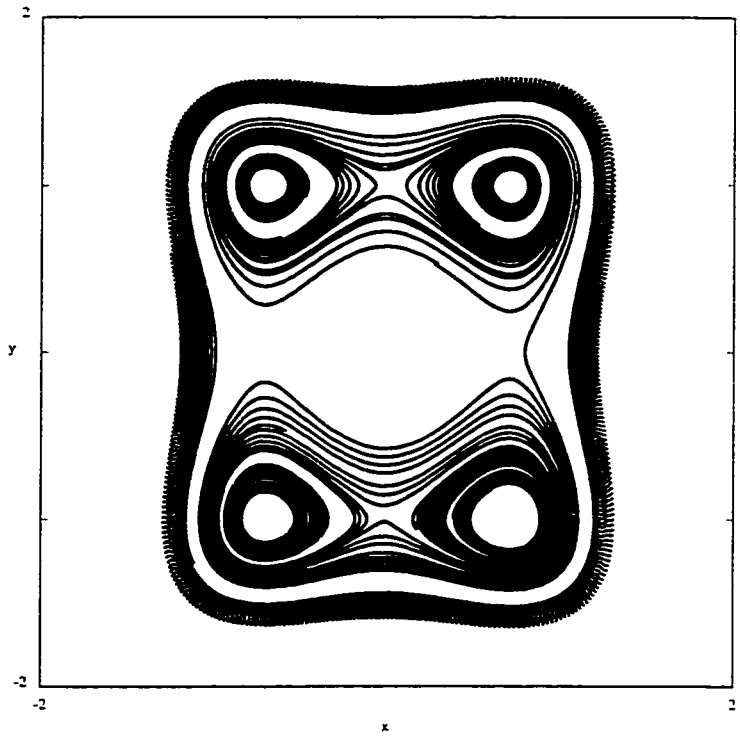

Figure 14: System (30) has eleven limit cycles for proper choice of parameters. Two limit cycles surround each of $A_1^0(1/\sqrt{2},1)$, $A_2^0(-1/\sqrt{2},1)$, $A_3^0(-1/\sqrt{2},-1)$ and $A_4^0(1/\sqrt{2},-1)$. One limit cycle surrounds each of the groups $A_1^0, A_2^0, S_1^0(0,1)$ and $A_3^0, A_4^0, S_2^0(0,-1)$. One limit cycle surrounds the group of all nine stationary solutions.

similar to well known uniqueness theorems. Unfortunately, quadratic systems never meet its conditions.

Theorem 5.12 *Consider the Liénard system*

$$\begin{aligned} \dot{x} &= y, \\ \dot{y} &= -f(x)y - g(x), \end{aligned} \tag{31}$$

or its equivalent system

$$\begin{aligned} \dot{x} &= y - F(x), \\ \dot{y} &= -g(x), \end{aligned} \tag{32}$$

where $f(x)$ and $g(x)$ are continuous for $x \in (a, b)$, $-\infty \leq b < 0 < a \leq \infty$, and $F(x) = \int_0^x f(s)ds$. Suppose there exists $x_1' \in (b, 0], x_1 \in [0, a)$ such that

- $g(x_1') = g(x_1) = 0$,
- $xg(x) > 0$ *for* $x \notin [x_1', x_1]$,
- $f(x) \leq 0 (\not\equiv)$ *for* $x \in (x_1', x_1) \cup \{0\}$.
- *functions* $f(x), g(x)/(x - x_1)$ *and* $(x - x_1)f(x)/g(x)$ *do not decrease in* (x_1, a).
- *functions* $f(x), g(x)/(x - x_1')$ *and* $(x - x_1')f(x)/g(x)$ *do not increase in* (b, x_1').

then system (31) has at most two limit cycles surrounding all critical points in $(b, a) \times \mathbb{R}$. If there are two limit cycles, then the exterior one is stable and inner one is unstable.

Even though system (29) is a rather special cubic system, there is no general result. Only a few specific cases of system (29) are well studied. The results for one such specific case are given below. In particular, Theorem 5.12 is applicable to it.

$$\begin{aligned} \dot{x} &= y - (a_3x^3 + a_1x), \\ \dot{y} &= -x(x^2 - 1) \end{aligned} \tag{33}$$

System (33) is integrable when $a_1 = a_3 = 0$ and it has no closed orbit or singular closed orbit when $a_1a_3 \geq 0$ and $a_1^2 + a_3^2 \neq 0$. Hence we need only to consider the case

$a_1 a_3 < 0$. Without loss of generality, we assume $a_1 < 0 < a_3$. Now, $O(0,0)$ is a saddle; $A(-1,0), B(1,0)$ are stable (unstable) antisaddles when $a_1 > -3a_3 (a_1 < -3a_3)$, and they are unstable weak foci of order 1 when $a_1 = -3a_3$.

Theorem 5.13 *For $a_3 > 0$, there exist functions $a_1 = a_{11}(a_3)$ $a_1 = a_{12}(a_3)$, where $a_1 = a_{11}(a_3)$ corresponds to a double loop bifurcation, for which system (33) has two singular closed orbits homoclinic to O and surrounding A, B, respectively; $a_1 = a_{12}(a_3)$ corresponds to a bifurcation of (large) limit cycle of multiplicity two with $-3a_3 < a_{11} < -\frac{5}{4}a_3$ and $a_{11} < a_{12} < -a_3$. As a_1 increases for fixed $a_3 > 0$, the limit cycle situation of system (33) is as follows*

- *For $a_1 \in (-\infty, -3a_3)$, there is no closed orbit around A and B alone, but there is a unique large limit cycle Γ_1 surrounding A, O, B and it is stable,*

- *For $a_1 \in (-3a_3, a_{11})$, Γ_1 is as above and there are exactly two limit cycles Γ_A, Γ_B bifurcated from A, B respectively, where $a_1 = -3a_3$ corresponds to Hopf bifurcation for both A and B (see Figure 15).*

- *As a_1 increases to some a_{11}, Γ_A, Γ_B expand and meet O simultaneously, and thus form two singular closed orbits (loops), while there exists a singular closed orbit Γ_0 surrounding Γ_A and Γ_B. For $a_1 > a_{11}$, Γ_0 breaks and bifurcates into a stable limit cycle surrounding all three critical points. The system then has two limit cycles surrounding all three critical points (see Figure 16),*

- *For $a_1 = a_{12}$, Γ_1 and Γ_2 coincide to form a semistable limit cycle. Thereafter a_1 is greater than a_{12} and no limit cycle surrounds all three critical points.*

Proof: The non-existence can be shown via a traditional Dulac function method. We also can use the uniqueness theorem for Liénard systems to show the uniqueness of the limit cycle here.

As for the conclusion concerning the existence of at most two limit cycles, we can apply the above theorem.

Let

$$g(x) = x^3 - x, \quad f(x) = 3a_3 x^2 + a_1, \quad F(x) = a_3 x^3 + a_1 x.$$

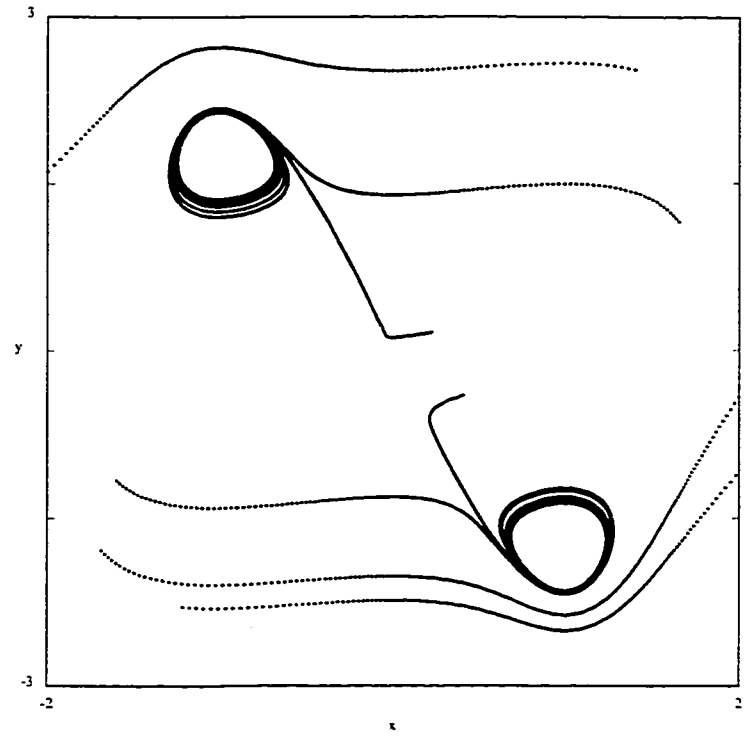

Figure 15: System (33) with $a_1 = -2.9, a_3 = 1$ has two limit cycles: one surrounds the upper-antisaddle: the other surrounds the lower-antisaddle

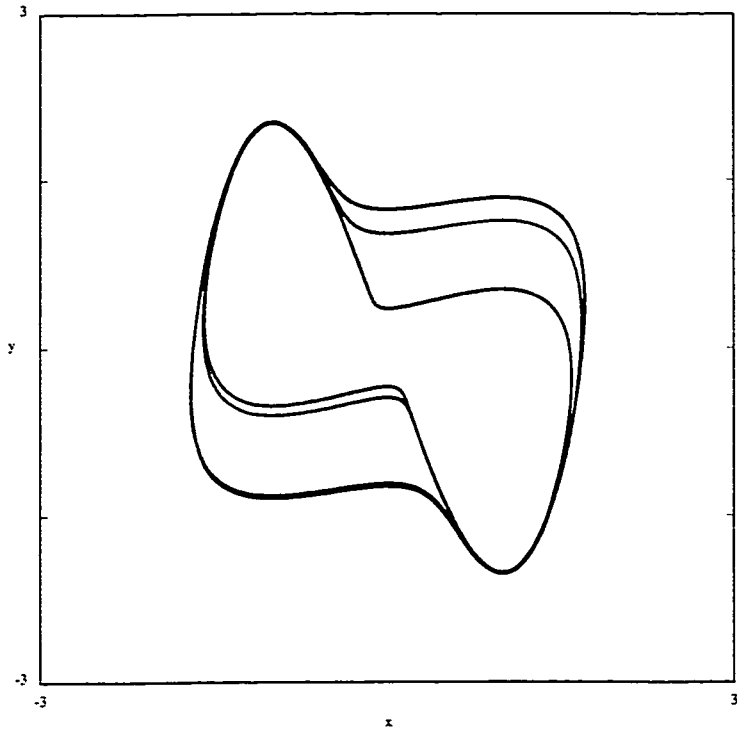

Figure 16: System (33) with $a_1 = -2.69, a_3 = 1$ has two limit cycles surrounding the group all three stationary solutions

The roots of $F(x) = 0$ and $f(x) = 0$ are

$$x_1 = -\sqrt{-\frac{a_1}{a_3}} \quad < \quad x_0 = 0 \quad < \quad x_2 = \sqrt{-\frac{a_1}{a_3}}.$$

and

$$\bar{x}_1 = \sqrt{-\frac{a_1}{3a_3}} \quad le \quad 0 < \quad \bar{x}_2 = \sqrt{-\frac{a_1}{3a_3}}.$$

respectively. The roots of $g(x) = 0$ are $-1, 0$, and 1. Since $a_1 < -a_3$, we have $x_1 < -1 < 0 < 1 < x_2$, and, clearly, $\bar{x}_1 \leq -1$, $\bar{x}_2 \geq 1$, for $-\infty < a_1 \leq -3a_3$; $-1 < \bar{x}_1 < 0 < \bar{x}_2 < 1$ for $-3a_3 < a_1 < -a_3$. It is easy to see that functions

$$3a_3x^2 + a_1, \quad x(x-1), \quad \frac{3a_3x^2 + a_1}{x(x-1)}$$

are monotonically decreasing for $-\infty < x < -1$ and the functions

$$3a_3x^2 + a_1, \quad x(x+1), \quad \frac{3a_3x^2 + a_1}{x(x+1)}$$

are monotonically increasing for $1 < x < \infty$. Therefore all conditions of the theorem are satisfied.

Open Problem: What is the maximum number of limit cycles for the Liénard system (29)?

Chapter 6

Bifurcations of Periodic Solutions for Quadratic Systems

As we have seen, Hilbert's 16th problem remains wide open. The traditional approaches, both algebraic theory and qualitative theory, have not been very successful. Thus it is natural to consider an alternative approach. Here we consider computer-assisted analysis. Conjecture 3 of Chapter 4 suggests that the maximum number of limit cycles for a given quadratic system equals its finite multiplicity m_f. This implies that any quadratic system has at most four limit cycles, in other words, $H(2) \leq 4$. But it is difficult to prove this conjecture. In this chapter, we carry out computer-assisted analysis for some interesting quadratic systems.

In Section 1, we study quadratic systems of finite multiplicity two and find that a semistable limit cycle can originate from the second order weak focus and terminates at a homoclinic loop. This supports that this class of quadratic systems has at most two limit cycles. It also shows that the homoclinic loop can bifurcate two limit cycles. In Section 2, we study quadratic systems of finite multiplicity four. We find that there are three limit cycles surrounding the origin for the famous Chen & Wang example, as expected. The computational results show that a semistable limit cycle of multiplicity two emanates from the origin, then meets the outer limit cycle and constitutes a limit cycle of multiplicity three at some point. The computations indicate that Chen & Wang example gives exactly four limit cycles. Our investigation therefore provides further evidence to support that quadratic systems have at most four limit cycles.

Recall that a quadratic system is an autonomous system of ordinary differential equations

$$\begin{aligned} \dot{x} &= a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2 \equiv P(x,y), \\ \dot{y} &= b_{00} + b_{10}x + b_{01}y + b_{20}x^2 + b_{11}xy + b_{02}y^2 \equiv Q(x,y). \end{aligned} \tag{34}$$

where $\dot{} \equiv d/dt$, $a_{ij}, b_{ij} \in R$, and $P(x,y), Q(x,y)$ are relatively prime real polynomials, of degree at most two, which are not both linear. In a way, quadratic systems may be considered to be the first class of systems away from linear systems. In fact, in the qualitative theory of ordinary differential equations, quadratic systems have been given ample attention, which led to a large number of papers on this subject

As discussed in Chapter 4, we can classify quadratic systems according to their finite multiplicities m_f. It seems natural for a systematic classification of phase portraits to start with the classes of lowest finite multiplicity and proceed to higher values of m_f. This is also a useful procedure because it permits to obtain information about a class of a higher finite multiplicity from that with a lower value of m_f by sending critical points back to the finite part of the plane and studying the bifurcation problems resulting from it.

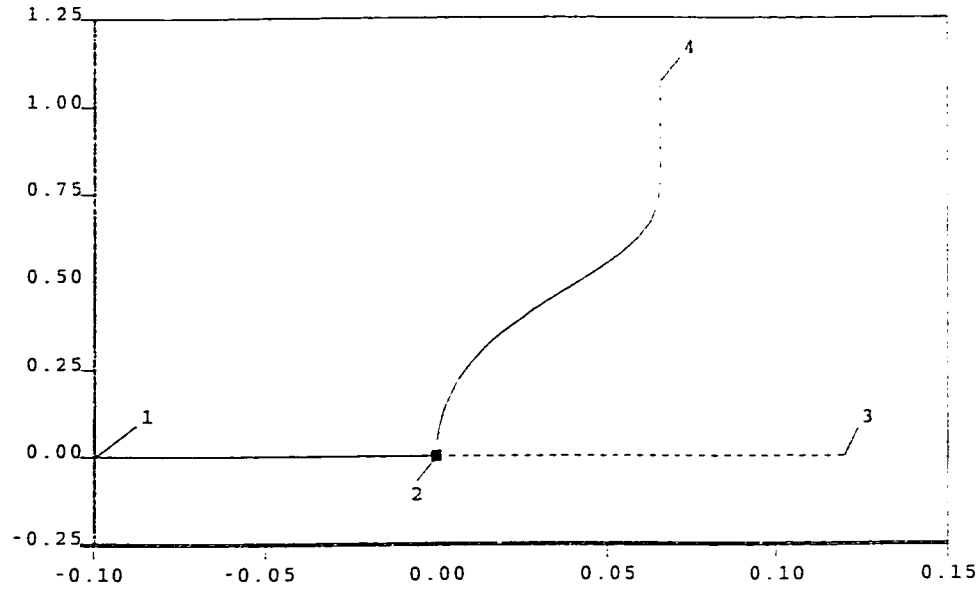

Figure 17: Periodic solution diagram of system (35) with $m = 1.5, l = 1$: L_2-norm versus d. Label 1 corresponds to a Hopf bifurcation and label 2 corresponds to a homoclinic loop

6.1 Quadratic systems of multiplicity two

Reyn & Kooij studied the phase portraits of quadratic systems of finite multiplicity two. They obtained 226 different phase portraits of nondegenerate quadratic systems with finite multiplicity two, and a classification in the $\mathbb{R}^{12}$ coefficient space. It is conjectured that in a quadratic system of this class there are at most two limit cycles.

A quadratic system with finite multiplicity two can have two real finite critical points, among which at most one can be a focus. Therefore, there exists at most one nest of limit cycles. Because we are only interested in limit cycles, we conclude the following lemma from the analysis in [49].

Lemma 6.3 *For a quadratic system of finite multiplicity two, if it has more than one limit cycle, then it can be transformed in the following form*

$$\begin{aligned} \dot{x} &= dx - y + lx^2 + mxy \\ \dot{y} &= x(1+x), \end{aligned} \tag{35}$$

where $d \in \mathbb{R}, l \in \mathbb{R}, m \in (0, +\infty)$.

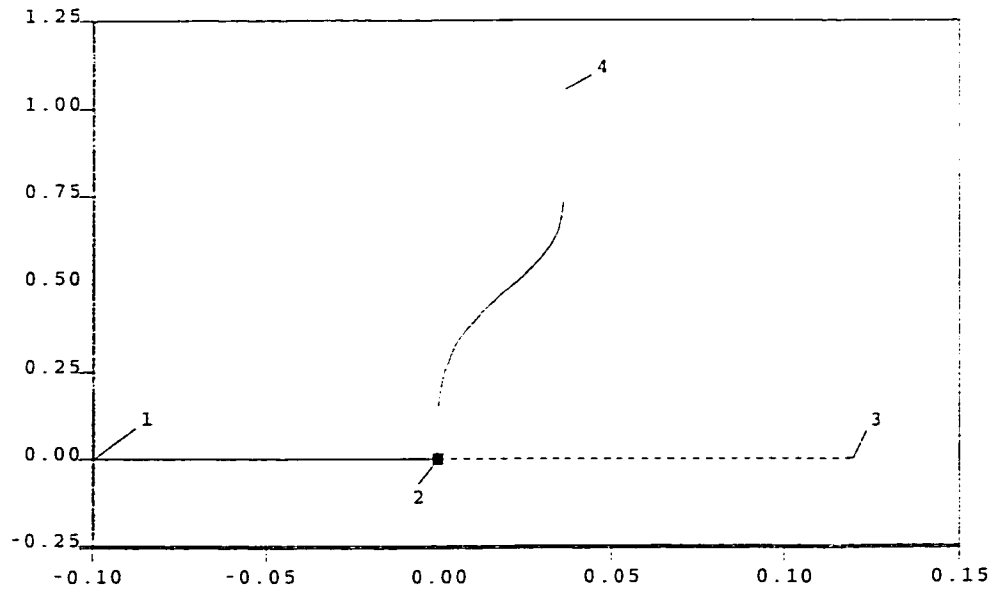

Figure 18: Periodic solution diagram of system (35) with $m = 2.0, l = 1$: L_2-norm versus d. Label 1 corresponds to a Hopf bifurcation and label 2 corresponds to a homoclinic loop

The above system is in the famous Ye Yanqian form, for which much work has been done. In particular, the equation belongs to class $(\mathrm{II})_{n=0}$([52]). It has been shown that system (35) with $0 < m < 2$ has at most one limit cycle. Thus the possible two-limit-cycle region is located in parameter space with $m > 2$. It remains unclear whether system (35) has at most two limit cycles.

Computations have been done for a range of parameters l, m. For $l = 1, m = 1.5$, the system has one stable limit cycle, where the diagram L_2-norm versus parameter d is given in Figure 17. For $l = 1, m = 2.0$, the system has one stable limit cycle, where the diagram L_2-norm versus parameter d is given in Figure 18 For $l = 1, m = 2.5$, the system has two limit cycles for $d \in (-0.00538, -0.0)$: the inner one is unstable: the outer one is stable (Figure 19). For $l = 1, m = 3.0$, the system has two limit cycles for $d \in (-0.016, 0.006)$: the inner one is unstable; the outer one is stable (Figure 20).

Furthermore, we also obtained the bifurcation of semistable limit cycles, m vs. b in Figure 21. Label 1, where $b = 0$ and $m = 1.0$, represents a weak focus of order two. Label 2 represents a perodic solution. Label 3 represents a homoclinic loop (see Figure 22). As expected, a limit cycle of multiplicity two emanates from a second

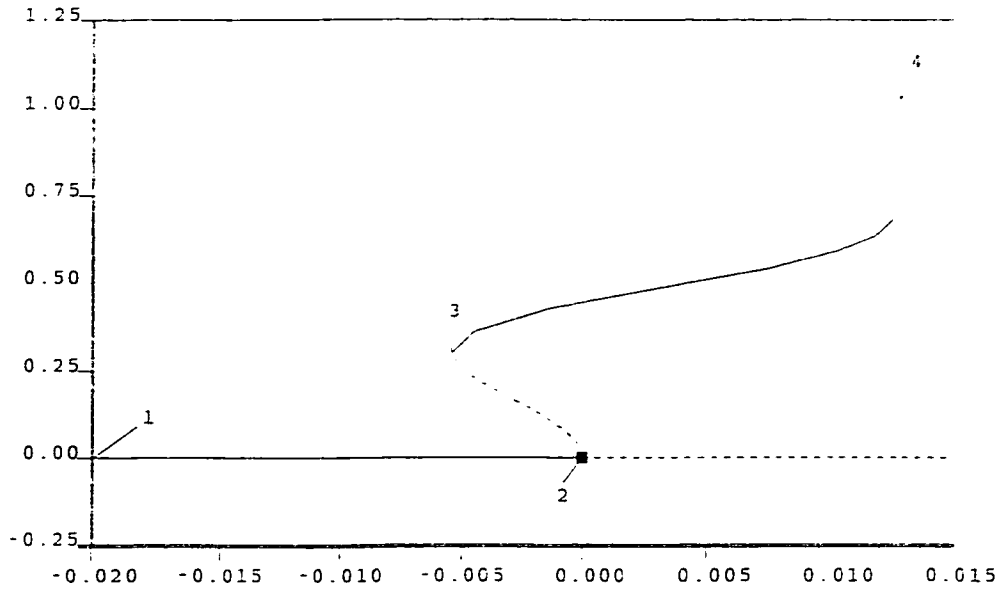

Figure 19: Periodic solution diagram of system (35) with $m = 2.5, l = 1$: L_2-norm versus d. Label 2 corresponds to a Hopf bifurcation: label 3 corresponds to a semistable limit cycle: and label 4 corresponds to a homoclinic loop

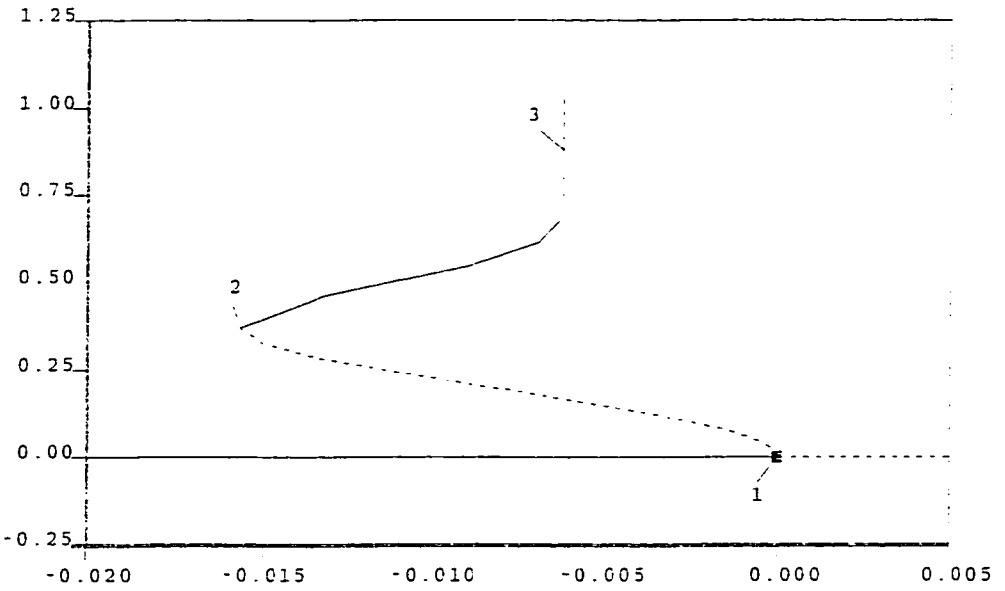

Figure 20: Periodic solution diagram of system (35) with $m = 3.0, l = 1$: L_2-norm versus d. Label 1 corresponds to a Hopf bifurcation: label 2 corresponds to a semistable limit cycle; and label 3 corresponds to a homoclinic loop

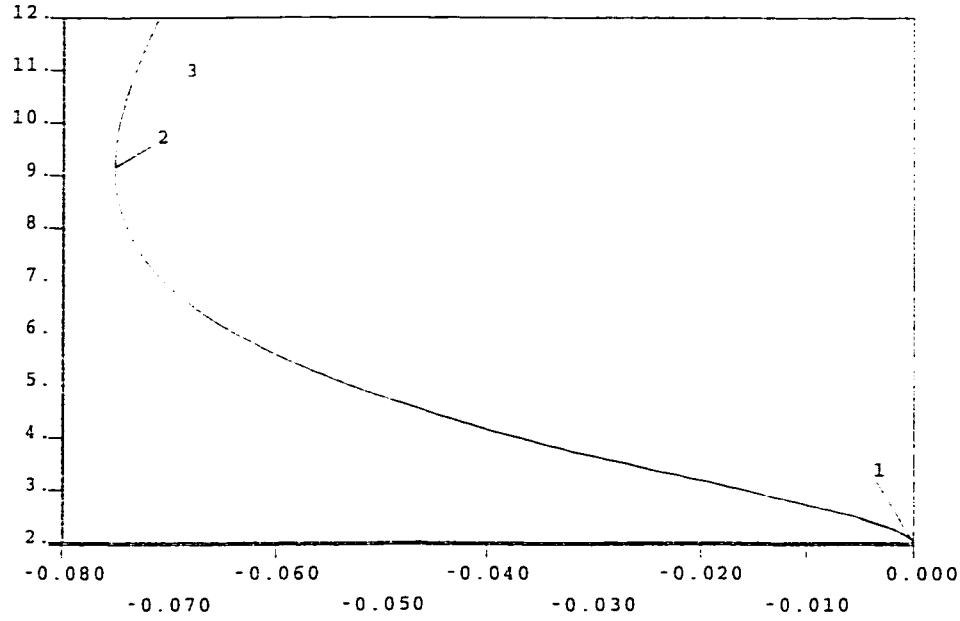

Figure 21: Bifurcation diagram of semistable limit cycles of system (35) : m vs. b. Label 1 represents a weak focus of order two, where $b = 0$ and $m = 1.0$. Label 3 represents the point where the system has a homoclinic loop

order weak focus, and continues in the two-parameter space m, b, finally approaching a homoclinic loop.

Thus our numerical analysis supports that system (35) as at most two limit cycles.

6.2 Quadratic systems of multiplicity four

Quadratic systems with finite multiplicity four are more complex than the previous cases. In fact, the examples of quadratic systems with four limit cycles are included in this class (See [6] and [53]). It is not known if these systems have at most four limit cycles, because the conclusion in [6] and [53] only suggest that these systems have at least four limit cycles.

The examples of quadratic systems with at least four limit cycles given by both Chen & Wang [6] and Shi [53] have two complex and two real critical points. It is conjectured that only quadratic systems with two complex and two real critical points can have four limit cycles. However, this conjecture is very difficult to prove. Actually, it is as difficult as proving that any quadratic system has at most four limit

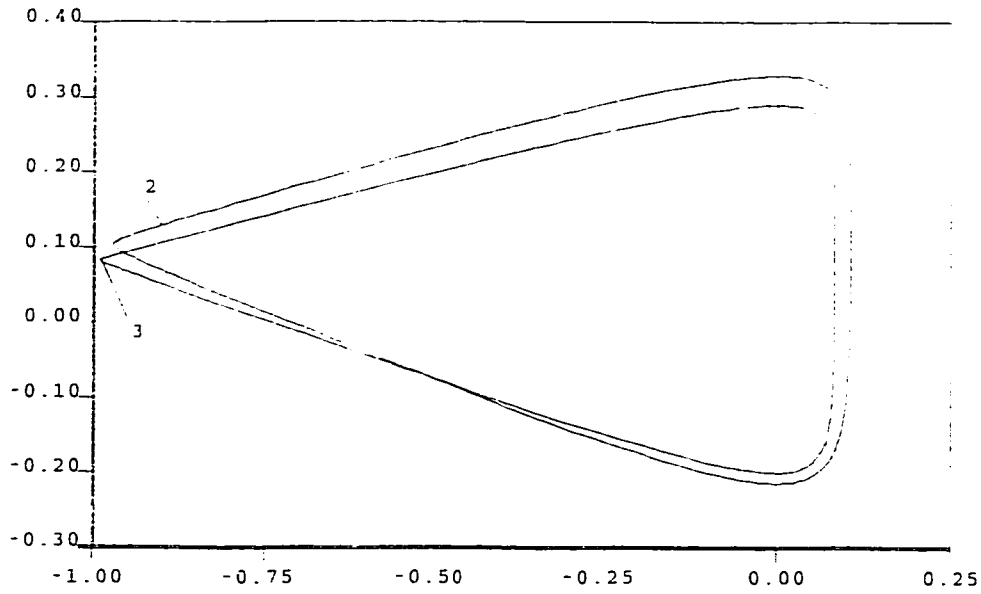

Figure 22: Periodic solution and homoclinic loop of system (35) : y vs. x. Label 2 indicates a periodic solution: label 3 indicates a homoclinic loop. The labels corresponds to those in Figure 21

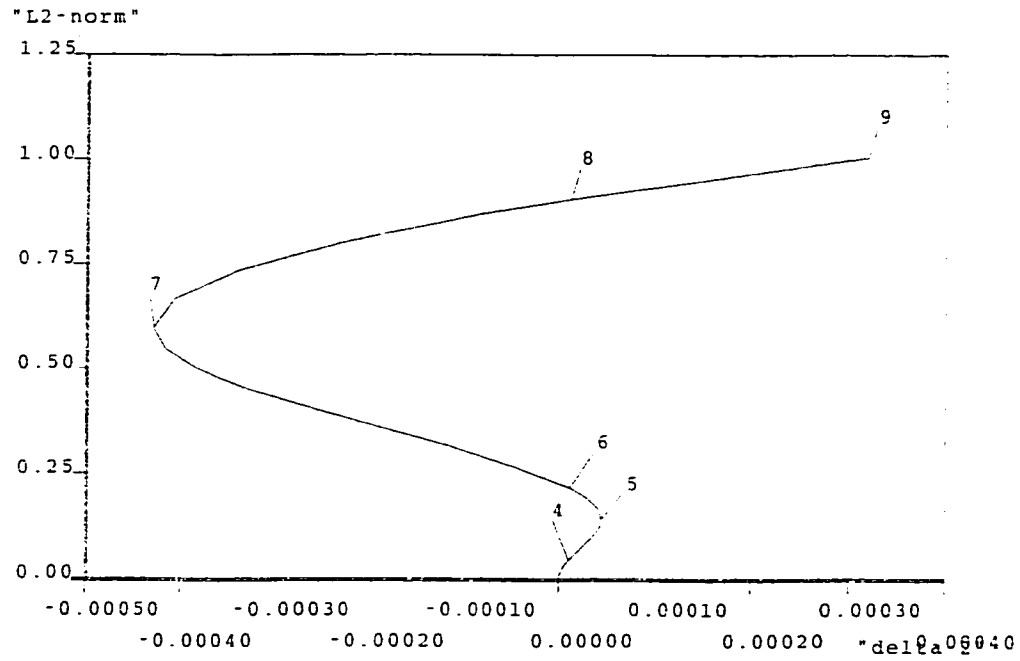

Figure 23: Periodic solution diagram of system (36) around $O(0.0)$ L_2-norm versus δ_2: $\delta_1 = 0.01$

cycles. Here we only conduct computer-assisted analysis for the typical case of four limit cycles. The evidence we present will offer further understanding of the limit cycle bifurcations.

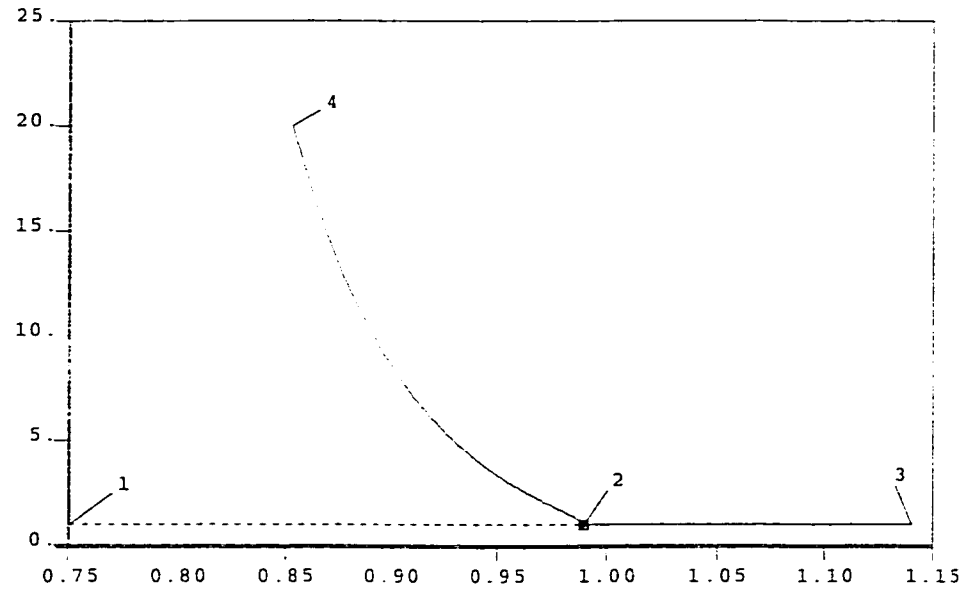

Figure 24: Periodic solution diagram of system (36) around $A(0,1)$: L_2-norm versus δ_2: $\delta_1 = 0.01$

Now we consider the examples of quadratic systems having four limit cycles. Chen & Wang [6] consider the quadratic systems

$$\begin{aligned} \dot{x} &= -\delta_2 x - y - 3x^2 + (1-\delta_1)xy + y^2. \\ \dot{y} &= x + \tfrac{2}{9}x^2 - 3xy. \end{aligned} \tag{36}$$

They showed that for $0 < \delta_2 << \delta_1 << 1$, system (36) has four limit cycles. More precisely, there are at least four limit cycles, since it was not proved that system (36) has exactly four limit cycles.

Lemma 6.4 *By choosing suitable small δ_1, δ_2, System (36) can have at least four limit cycles, distributed in two nests: one nest contains at least three limit cycles; the other contains at least one. ([6])*

The famous examples above are obtained by perturbing quadratic systems with a second order weak focus. The estimate is given for small parameters.

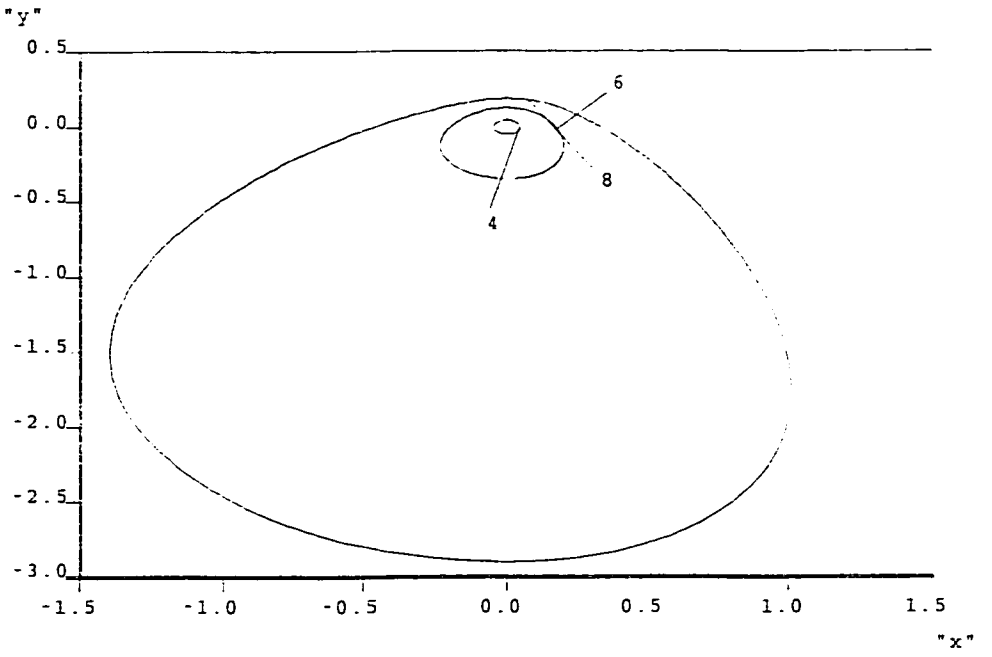

Figure 25: Periodic solutions labeled 4.6.8 in Figure 23. For $\delta_1 = 0.01$ and $\delta_2 = 0.00001$ the system has three limit cycles surrounding the origin $O(0.0)$

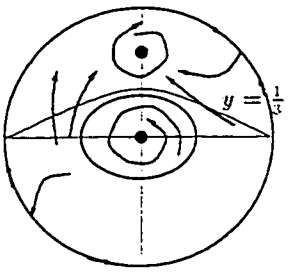

Figure 26: Phase diagram of system (36) on the Poincaré sphere

We first give a diagram of three limit cycles surrounding the origin of system (36). Then we present its semistable limit cycle diagram. By parameter continuation, fixing $\delta_1 = 0.01$, we have a periodic solution diagram: L_2-norm versus δ_2 as shown in Figures 23 and 24, where periodic solutions labeled 4, 6, 8 are given in Figure 25. Qualitative analysis guarantees that there exists at least one limit cycle around $A(0,1)$. Figure 24 shows that this limit cycle is very large. We can not get a limit cycle around $A(0,1)$ numerically. But qualitative analysis, referring to Figure 26, shows that $A(0,1)$ is unstable and that the area bounded by $y = \frac{1}{3}$ and upper half sphere constitutes the region of attraction. Thus, by Bendixson's theorem, there must be at least one limit cycle around $A(0,1)$. Our numerical analysis supports that system (36) has four limit cycles for $\delta_1 = 0.01$ and δ_2 in a fine interval around $\delta_2 = 0$. We also obtained a global bifurcation diagram for periodic solutions. Moreover, Figures 27 and 28 give bifurcation diagrams of limit cycles with multiplicity two. Here we trace the fold bifurcation in parameters δ_1, δ_2, i.e., the semistable limit cycle bifurcation of system (36). The δ_1 versus δ_2 diagram is given in Figures 27 and 28. We see a cusp point at label 10 , which corresponds to a limit cycle of multiplicity three.

Again, these results are as expected. Only three limit cycles are found to surround the focus. All three limit cycles surround the origin, then no more than one limit cycle surrounds the other focus.

In summary, our computer-assisted analysis supports that

The quadratic system (36) has at most four limit cycles.

Similarly, we can also explore limit cycle bifurcations for more general quadratic systems. With computer-assisted analysis, one can uncover information which can not be obtained with qualitative theory, and one can obtain evidence towards an answer to Hilbert's 16th problem.

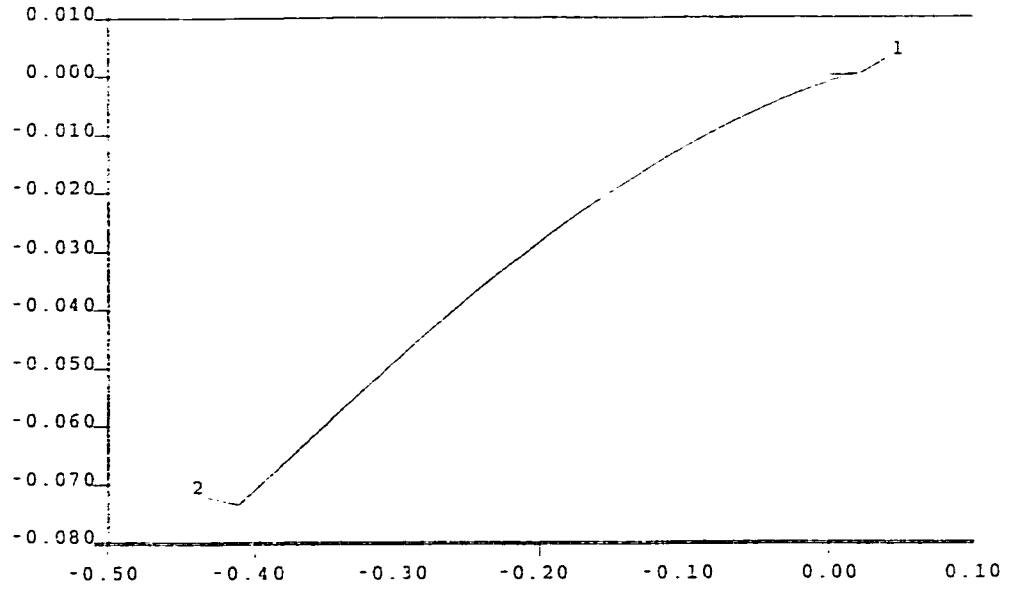

Figure 27: Fold bifurcation diagram, i.e. semistable limit cycle bifurcation curve of system (36): δ_1 versus δ_2, and a cusp at label 1, which corresponds to a limit cycle with multiplicity three

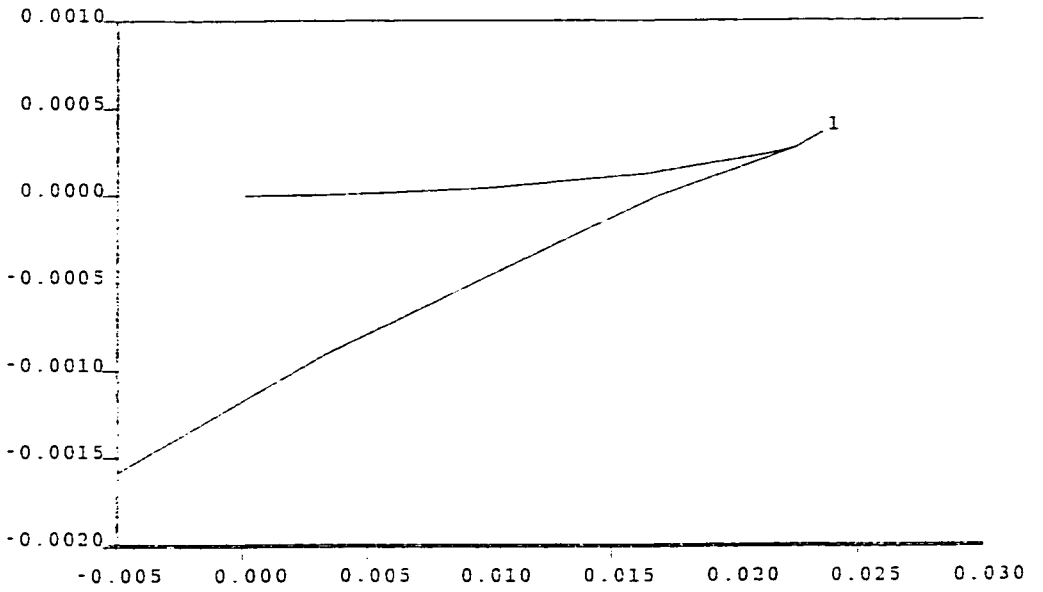

Figure 28: Fold bifurcation diagram: an enlarged view of Figure 27. Label 1 corresponds to a limit cycle of multiplicity three. It is easy to see that the semistable limit cycle bifurcation starts from $\delta_1 = 0, \delta_2 = 0$.

Chapter 7

Bifurcations of Periodic Solutions for Cubic Systems

Now we will turn away from the core of Hilbert's 16th problem, because studies of cubic systems are far from solving the Hilbert 16th problem for $n = 3$. The cubic Liénard systems of this chapter are derived when codimension-three bifurcations are investigated ([15]). They are related to Hilbert's 16th problem in the sense that we can see how bifurcations of limited cycles contribute to the number of limit cycles. The study of these systems is not completed, in particular, there is no full knowledge about limit cycles when a cusp presents or when there are two saddles. In this chapter, we study two cases of cubic Liénard systems corresponding to the focus/elliptic case and the saddle case of codimension-three bifurcations, which are extracted from ([15]). The detailed analysis is omitted here. For the first case, the cubic Liénard system has one cusp and one anti-saddle. The computations indicate that such a system can have a singular closed orbit homoclinic to the cusp and a limit cycle surrounding it (Section 1). For the second case, the cubic Liénard system has two saddles and one anti-saddle. The computation indicates that this system can have two limit cycles for certain parameters (Section 2).

We study the bifurcation of the cubic Liénard system

$$\begin{aligned} \dot{x} &= y, \\ \dot{y} &= G(x) + yf(x), \end{aligned} \tag{37}$$

where $F(x)$ is real polynomial of x with degree three and $f(x)$ is real polynomial of x with degree two. It is easy to see that system (37) is equivalent to system (29). Recall that we have studied the special case of system (37) in Chapter 5, which corresponds to system (37) with $G(x) = x(1-x^2)$, and we have obtained very satisfactory results. However, for general case, we are unable to achieve the similar results. Thus we have to turn to other tools. A systematical numerical study is an option. With five free parameters, this is a difficult task. Here we use parameter continuation to explore some special bifurcations. We shall see that some of results obtained are beyond the current scope of qualitative theory.

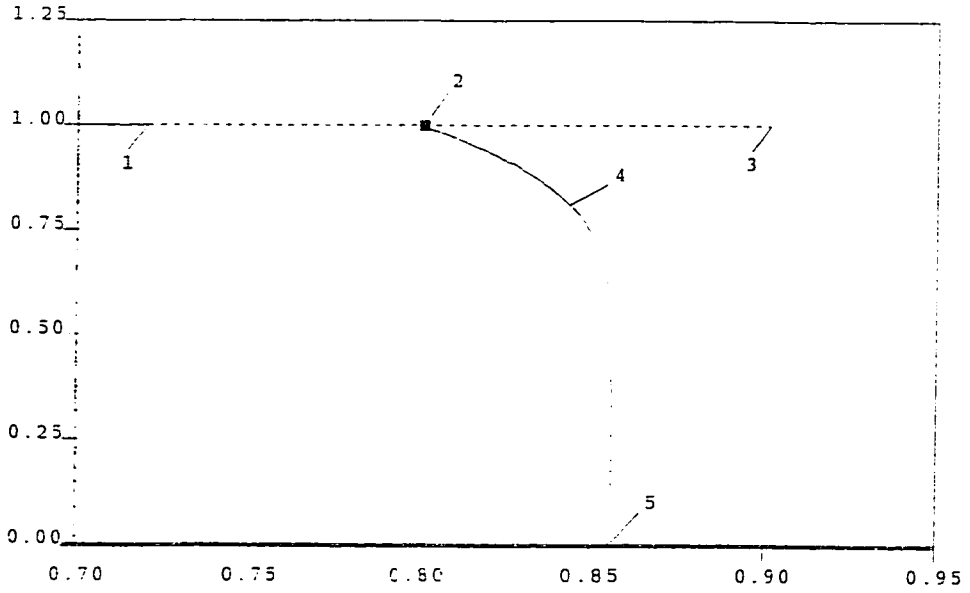

Figure 29: Periodic solution diagram of system (37) L_2-norm versus r_5: Case I with $r_3 = 1.0, r_4 = -.8$, where label 2 corresponds to the Hopf bifurcation point and label 5 corresponds to a homoclinic loop through the cusp point

7.1 The cubic Liénard system: case I

First we study system (37) with

$$G(x) = r_3x^2 - x^3; f(x) = r_4x + r_5x^2.$$

This corresponds to the focus/elliptic case of codimension-three bifurcations in [15].

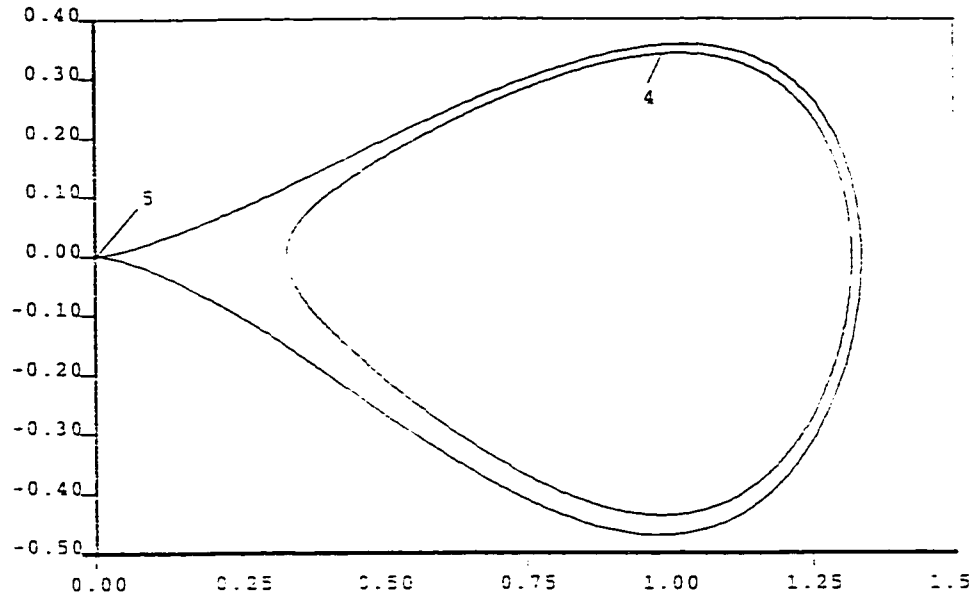

Figure 30: Periodic solution diagram of system (37) y vs. x: Case I with $r_3 = 1.0, r_4 = -.8$, where label 4, 5 correspond to those in Figure 29

Fixing $r_3 = 1.0, r_4 = -.8$, and letting r_5 be a free parameter, we get a system having $O(0,0)$ as a cusp and $A(r_3, 0)$ as a elementary critical point, when $r_3 \neq 0$. Numerical computations show the periodic solution curve vs. parameter r_5, as given in Figure 29. Selected periodic solutions are given in Figure 30. In particular, we see that the system has a singular closed orbit passing through the cusp $O(0,0)$. Furthermore, phase diagrams show the following: 1) When $r_5 = .9$, system (37) is globally asymptotically stable (see Figure 31). 2) When $r_5 = .855$, system (37) has a singular closed orbit passing through the cusp $O(0,0)$ and, in addition, a stable periodic solution surrounds this orbit (see Figure 32). 3) When $r_5 = .75$, the system (37) has two limit cycles, the outer one is stable and the inner one is unstable (see Figure 33).

In System (37), let $G(x) = r_3x^2 - x^3$, $f(x) = r_4x + r_5x^2$. When $r_3 = 1.0$, $r_4 = -.8$, and r_5 is free, the periodic solution L_2-norm vs. parameter r_5 is given in Figure 29. In particular, the system can have a singular closed orbit going through cusp: $O(0,0)$ and a limit cycle simultaneously.

7.2 The cubic Liénard system: case II

Now we study system (37) with

$$G(x) = -x + x^3; \ f(x) = b_1 + x + b_2x^2.$$

This corresponds to the saddle case of codimension-three bifurcations in [15].

It is easy to see that system (37) has three critical points: one node $O(0,0)$ and two saddles: $A(-1,0)$, $B(1,0)$.

We choose $b_2 = .3$ and let b_1 be free. We obtain the periodic solution vs. parameter b_1, as given in Figure 34. Selected periodic solutions are given in Figure 36 and the homoclinic loop through $A(-1,0)$ is given in Figure 37.

In Summary, we have the following

Let $G(x) = -x + x^3$; $f(x) = b_1 + x + b_2x^2$ in System (37). When $b_2 = .3$ and b_1 free, the periodic solution vs. parameter b_1 is given in Figure 34. In particular, the system can have a singular closed orbit passing through the saddle: $A(-1,0)$. Label 2 corresponds to the Hopf bifurcation point. Label 6 corresponds to a heteroclinic cycle. Label 4 and 5 correspond to $b_1 = -0.0435$, so the system has two limit cycles.

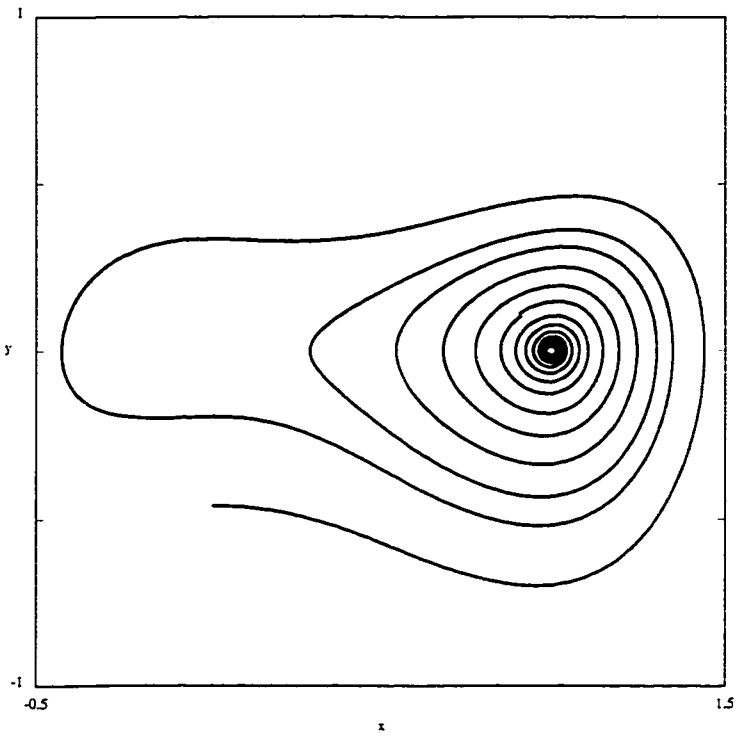

Figure 31: Phase diagram of system (37): Case I with $r_3 = 1.0, r_4 = -.8, r_5 = .9$. The system is globally asymptotically stable.

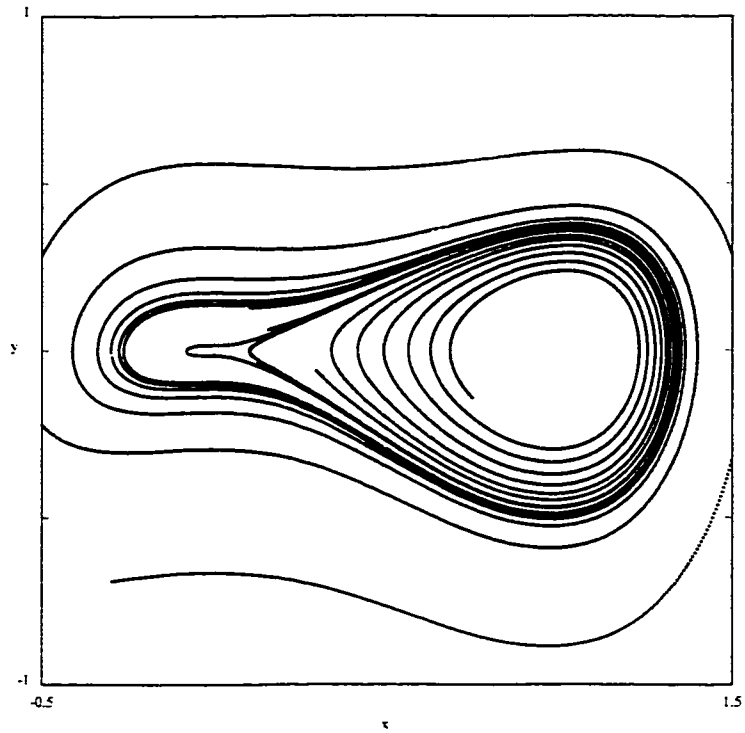

Figure 32: Phase diagram of system (37): Case I with $r_3 = 1.0, r_4 = -.8, r_5 = .855$. The system has a singular closed orbit going through the cusp $O(0,0)$ and, simultaneously, a limit cycle surrounding it.

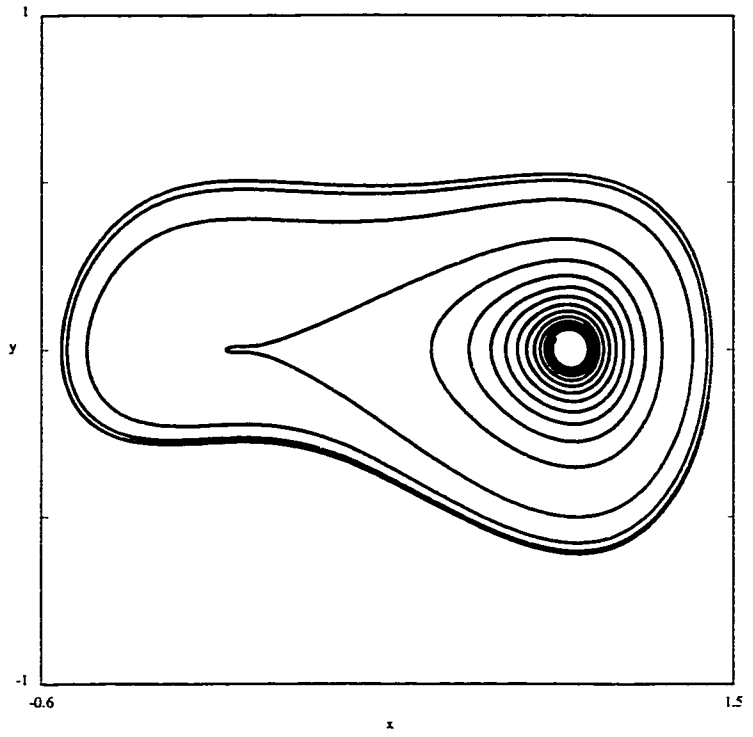

Figure 33: Phase diagram of system (37): Case I with $r_3 = 1.0, r_4 = -.8, r_5 = .75$. The system has two limit cycles: the outer one is stable and the inner one is unstable.

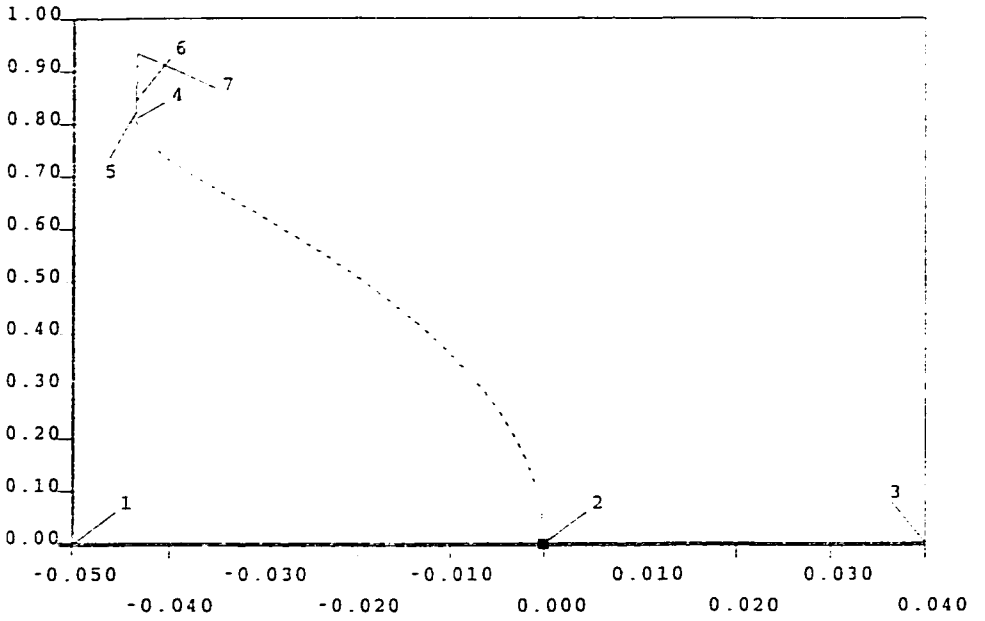

Figure 34: Periodic solution diagram of system (37), L_2-norm versus b_1: Case II with $b_2 = .3$. Label 2 corresponds to the Hopf bifurcation point. Labels 4 and 6 correspond two periodic solutions for $b_1 = -0.0435$. and Label 5 corresponds to a semistable periodic solution. Label 7 corresponds to a heteroclinic cycle.

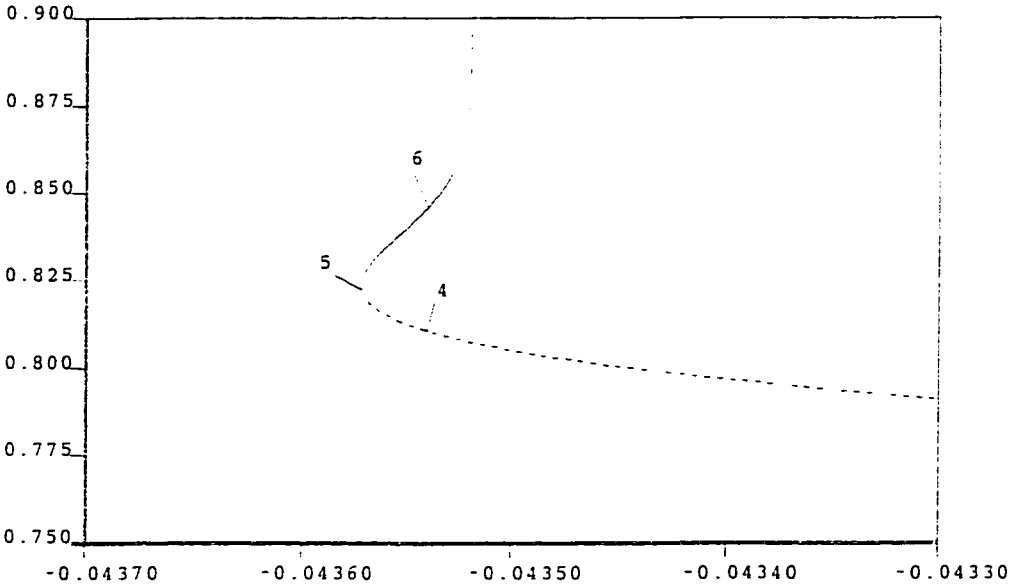

Figure 35: An enlarged view of Figure 34. Labels 4 and 6 correspond two periodic solutions for $b_1 = -0.0435$. and Label 5 corresponds to a semistable periodic solution.

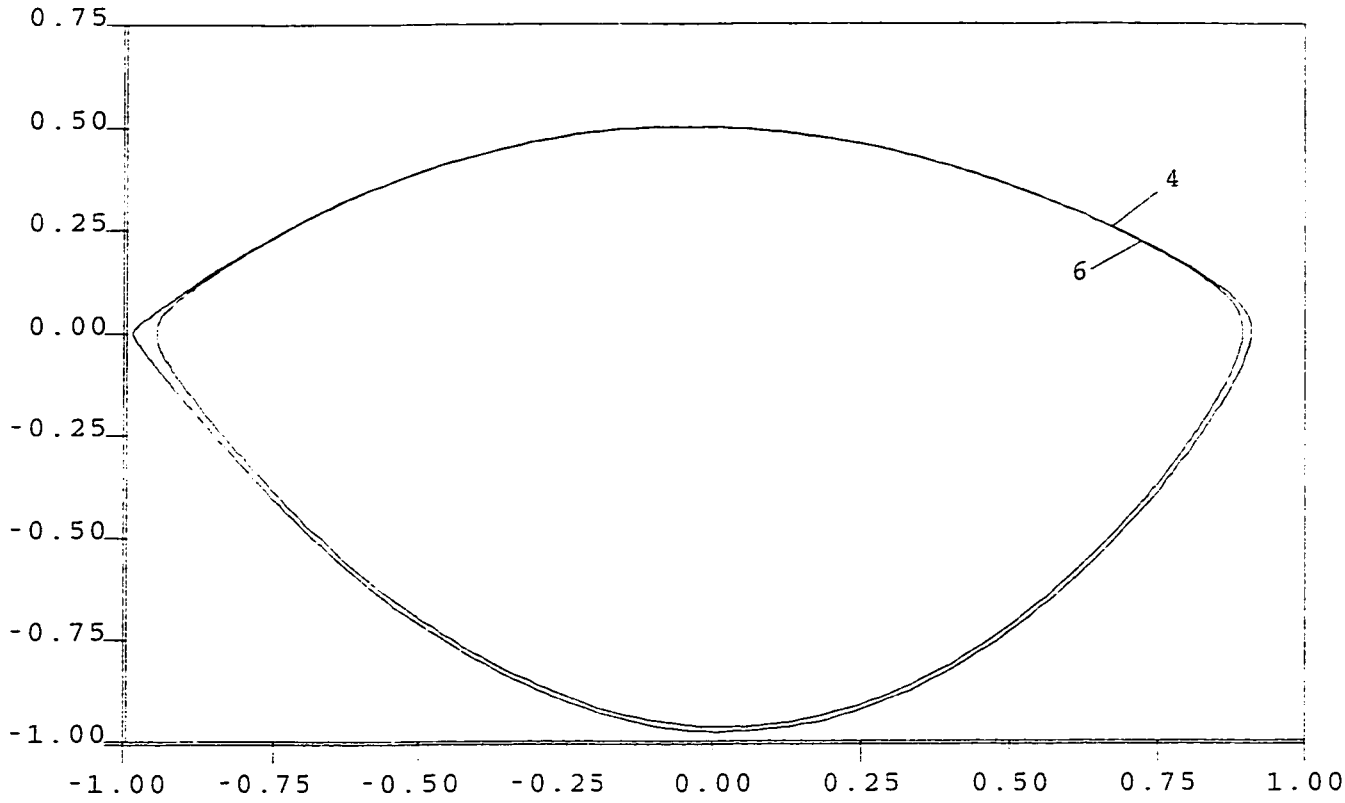

Figure 36: Periodic solution diagram of system (37): y vs. x. Case II with $b_2 = .3$. where labels 4, 6 correspond to those in Figure 34. This means that for $b_1 = -0.0435$ and $b_2 = .3$. the system has two limit cycles

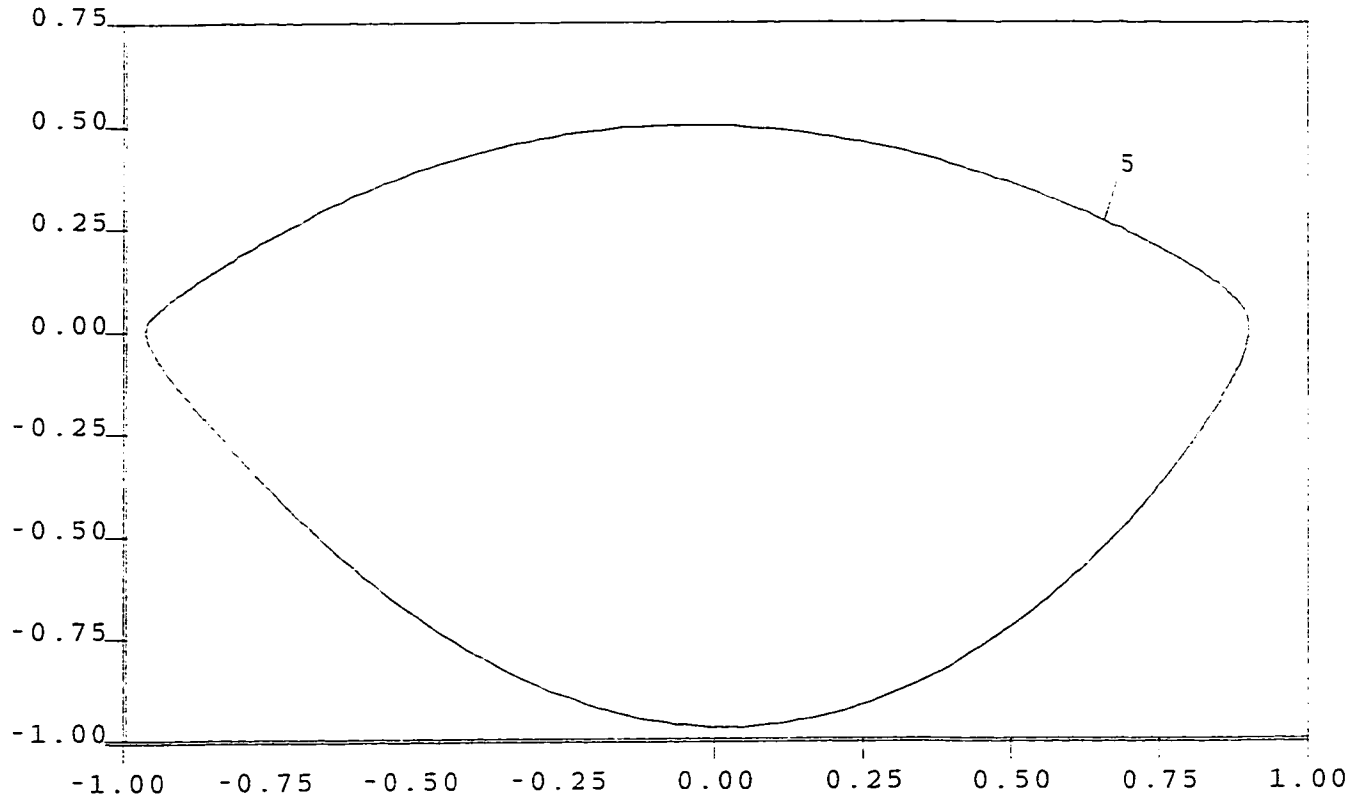

Figure 37: A semistable periodic solution of system (37): y vs. x. Case II with $b_2 = .3$, where label 5 corresponds to those in Figure 34

Appendix A

A Brief User Manual of QSYS (Java 1.1)

QSYS is a Java program to visualize quadratic systems.

$$\begin{aligned} \dot{x} &= a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2, \\ \dot{y} &= b_{00} + b_{10}x + b_{01}y + b_{20}x^2 + b_{11}xy + b_{02}y^2. \end{aligned}$$

QSYS provides an easy-to-use graphical user interface (GUI) (See Figure 38). The window is divided into two parts: **Entry Area** (*the up part of the interface*) and **Graphics Area** (*the low part of the interface*). **Entry Area** is used to enter values of coefficients of quadratic systems and computing control variables. **Graphics Area** is used to display orbits of a given quadratic system.

QSYS works as both *applet* and *stand-alone* applications under *Java 1.1*.

A.1 Getting Started with Stand-alone QSYS

1) Enter **java QSYS** in Unix prompt. The **QSYS** main window (see Figure 38) will appear.

2) Input desired values for coefficients and control variables in **Entry Area**.

3) Select **File→Start**. This is to get the program ready and helps to verify the equations.

4) Double-Click in **Graphics Area** to choose initial points of orbits. Select **Edit→Delete Last Orbit** to remove the most recent orbit if desired.

5) Select **File→Print** to produce a hard copy; or Select **File→Save** to save a phase diagram into a QSYS data file.

A.2 QSYS Commands

A.2.1 Mouse Commands

Double Click: By Double-Clicking in the **Graphics Area**, the program sets the mouse current position as an initial point and draws an orbit from it in $(MinX, MaxX) \times (MinY, MaxY)$.

A.2.2 Menu Commands

There are three menu bars: **File, Edit, Screen** and **Help**. **File** has five items: **Load, Save, Print, Clear** and **Quit**. **Edit** has two items: **Remove Last Orbit, Start Over**. **Screen** has two items: **Clear Screen** and **Fresh Screen**. **Help** has only one item: **About**.

File → **Start** (Control-G): It is a start point of the program. When it is selected, internal data will cleared and the system get ready to work.

→ **Load** (Control-L): When it is selected, a file dialog window is poped up and asks for a file name. This file must be a **QSYS** data file; otherwise, a dialog wondow appears to give hints. When a **QSYS** data file is loaded, the saved data is displayed graphically.

→ **Save** (Control-S): When it is selected, a file dialog window is poped up and asks for a file name. The system will save the **QSYS** data into this file.

→ **Print** (Control-P): When it is selected, a printing dialog window is poped up. A user may choose either *Print to Printer* or *Print to File*. When a user clicks *Printer*, he needs to enter a printing command of the operating system. When a user clicks *File*, he needs to enter a file name, to which a postscript file will be saved.

→ **Quit** (Control-Q): When it is selected, the program quits.

Edit → **Delete Last Orbit** (Control-R): When it is selected, the most recent orbit is deleted.

→ **Delete All** (Control-M): When it is selected, phase drawing is restarted.

Screen → **Clear Screen** (Control-K): When it is selected, the screen is cleaned and the internal data remains.

→ **Fresh Screen** (Control-F): When it is selected, the screen is updated according to the internal data.

Help → **About** (Control-A): When it is selected, a message window is poped up and information about this program is displayed.

A.2.3 Button Commands

1) **Title** button: Title "Quadratic System" is a start/clear button. When it is pressed, the program is set to start or restart.

2) **Delete Last Orbit** button: When it is pressed, the last drawed or bit is deleted. (It appears when the program works as applet only.)

A.3 Coefficients of Quadratic Systems

Equations of Quadratic Systems are given graphically with coefficients: a_{00}, a_{10}, a_{01}, a_{20}, a_{11}, a_{02}, b_{00}, b_{10}, b_{01}, b_{20}, b_{11}, and b_{02}. They are defined as float numbers. User can enter the desired coefficient values of the equation through **Entry Area**. Their ranges are limited between -9999 and 9999.

A.4 QSYS Control Variables

Control variables are also given graphically. User can enter the desired values of control variables through **Entry Area**.

Ranges can be adjusted by changing **Minimum X**, **Maximum X**, **Minimum Y**, and **Maximum Y**. Horizontal ranges are between **Minimum X** and **Maximum X**; vertical ranges are between **Minimum Y** and **Maximum Y**. They are defined as float numbers.

Step Size defines the integral step. It can be either a positive *float number* or a negative one in $(-.4..4)$. A user may adjust **Step Size** per orbit.

Step# defines the number of integral steps. It must be a *positive integer* and between 10 and 9999. **Step#** can be adjusted per orbit.

Note: 1) there must hold that: *Minimum X* is less than *Maximum X* , and *Minimum Y* is less than *Maximum Y*. 2) All of *Minimum X*, *Maximum X*, *Minimum Y*, and *Maximum Y* must be between -9999 and 9999.

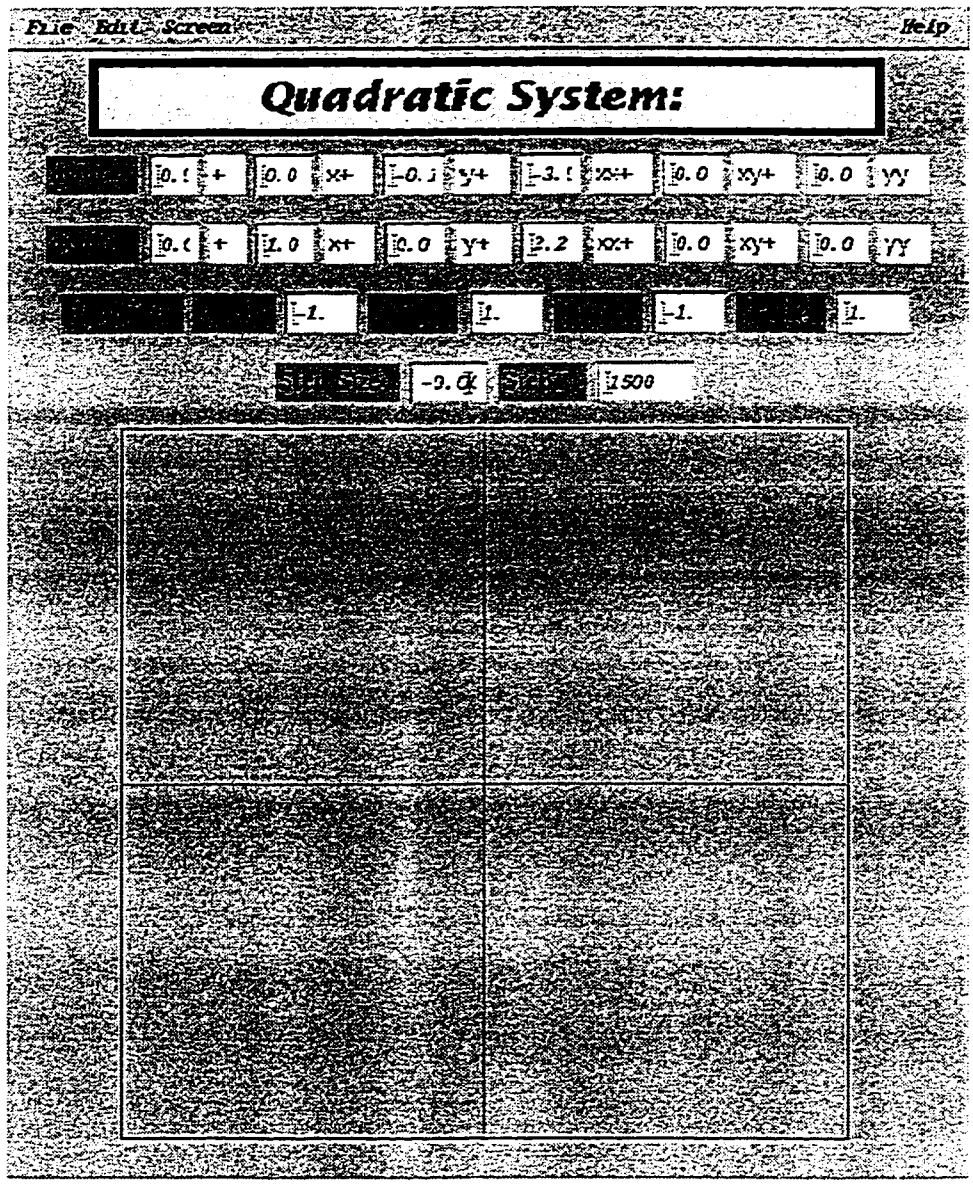

Figure 38: The graphical user interface (GUI) of QSYS (Java 1.1)

Appendix B

Java Source Code of QSYS

B.1 QSYS.java

```
//////////////////////////////////////////////////////////////////////////
//// Quadratic System phase drawing (under Java 1.1)                  /////
////                                                                  /////
//// dx/dt=a_00 +a_10 x +a_01 y +a_20 xx+ a_11 xy +a_02 yy            /////
//// dy/dt=b_00 +b_10 x +b_01 y +b_20 xx+ b_11 xy +b_02 yy            /////
/////                                                                /////
/////By Xianhua Huang                                               /////
/////Nov. 6 1998                                                    /////
/////At Concordia University                                        /////
/////   Montreal                                                     /////
/////   Canada                                                       /////
//////////////////////////////////////////////////////////////////////////
//////Basic functions:                                                 /////
//////    1) Double click: choose an initial (x, y) then draw an       /////
/////        orbit from it                                            /////
//////    2) coefficients, scales, step size and step number           /////
/////         can be entered from the entry fields                      /////
/////     3) Menu functions: Start, Load, Save, Print,  Quit,          /////
/////         Clear Screen, Fresh Screen                              /////
//////////////////////////////////////////////////////////////////////////
```

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.lang.*;
import java.util.*;
import java.io.*;


public class QSYS extends Applet implements MouseListener, ActionListener  {

private static  Frame frm = new Frame("Quadratic System");  // Create a window

/*****************************************/
/***main(): For stand alone applications***/
/*****************************************/
 public static void main(String[] args) {
    QSYS  qds = new QSYS();   // Create the applet panel
    frm.add(qds, "Center");     // Add applet to window
    qds.init();                 // Initialize the applet

//////////////////////////////////////////////////
// Create a menubar and tell the frame about it//
    MenuBar menubar = new MenuBar();
    frm.setMenuBar(menubar);

//////////////////////////////////////////////////
// Create three pulldown menus for the menubar//
    Menu file = new Menu("File");
    Menu edit = new Menu("Edit");
    Menu screen = new Menu("Screen");
    Menu help = new Menu("Help");

// Add the menus to the bar, and treat Help menu specially.
```

```
    menubar.add(file);
    menubar.add(edit);
    menubar.add(screen);
    menubar.add(help);
    menubar.setHelpMenu(help);

// Add item "Start"
    MenuItem start = new MenuItem("Start", new MenuShortcut(KeyEvent.VK_G));
    start.addActionListener( qds);    // Say who's listening for the events
    start.setActionCommand("start"); // A detail to go along with the events
    file.add(start);                  // Add item to menu pane

// Add item "Open"
    MenuItem open = new MenuItem("Load", new MenuShortcut(KeyEvent.VK_L));
    open.addActionListener(qds);     //Say who's listening for the events
    open.setActionCommand("open");   // A detail to go along with the events
    file.add(open);

// Add item "save"
    MenuItem save = new MenuItem("Save", new MenuShortcut(KeyEvent.VK_S));
    save.addActionListener(qds);     //Say who's listening for the events
    save.setActionCommand("save");   // A detail to go along with the events
    file.add(save);

// Add item "Print"
    MenuItem print = new MenuItem("Print", new MenuShortcut(KeyEvent.VK_P));
    print.addActionListener(qds);      //Say who's listening for the events
    print.setActionCommand("print");   // A detail to go along with the events
    file.add(print);

//Add item "Quit"
    MenuItem quit = new MenuItem("Quit", new MenuShortcut(KeyEvent.VK_Q));
    quit.addActionListener( qds);
    quit.setActionCommand("quit");
```

```
    file.add(quit);

//Add items to Edit menu
    MenuItem removelast =new MenuItem("Delete Last Orbit",
                         new MenuShortcut(KeyEvent.VK_R));
    removelast.addActionListener( qds);
    removelast.setActionCommand("rmlast");
    edit.add(removelast);

MenuItem removeall =new MenuItem("Delete All",
                    new MenuShortcut(KeyEvent.VK_M));
    removeall.addActionListener( qds);
    removeall.setActionCommand("rmall");
    edit.add(removeall);


//Add items to Screen menu: Clear Screen, Fresh Screen
 MenuItem clscreen =new MenuItem("Clear Screen",
                    new MenuShortcut(KeyEvent.VK_K));
 clscreen.addActionListener( qds);        //Say who's listening for the events
 clscreen.setActionCommand("clearscreen");//A detail to go along with the event
 screen.add(clscreen);

 MenuItem frscreen =new MenuItem("Fresh Screen",
                    new MenuShortcut(KeyEvent.VK_F));
 frscreen.addActionListener( qds);     // Say who's listening for the events
 frscreen.setActionCommand("fresh");   // A detail to go along with the events
 screen.add(frscreen);


// Add items to the help menu,
  MenuItem about = new MenuItem("About", new MenuShortcut(KeyEvent.VK_A));
  about.addActionListener( qds);
  about.setActionCommand("about");
```

```
  help.add(about);

//Define layout
    frm.setSize(600, 760);                    // Set the size of the window
    frm.setTitle("Quadratic System");
    frm.show();                               // Make the window visible
    frm.addWindowListener(new WindowAdapter() {// Handle window close requests
      public void windowClosing(WindowEvent e) { System.exit(0); }
    });
  }
/**********************/
/****End of main()****/
/**********************/

 /*************************************************************************
   * This is the convenience routine for adding menu items to a menu pane.
   * It works for pulldown or popup menu panes, since PopupMenu extends Menu.
   *************************************************************************/
  protected static void createMenuItems(Menu pane, ActionListener listener,
                                        String[] labels, String[] commands,
                                        int[] shortcuts) {
    for(int i = 0; i < labels.length; i++) {
      MenuItem mi = new MenuItem(labels[i]);
      mi.addActionListener(listener);
      if ((commands != null) && (commands[i] != null))
        mi.setActionCommand(commands[i]);
      if ((shortcuts != null) && (shortcuts[i] != 0))
        mi.setShortcut(new MenuShortcut(shortcuts[i]));
      pane.add(mi);
    }
  }

/*************************************************************************
   * This is the method defined by the ActionListener interface.  All
```

```
   * the menu item commands are handled here because the applet was specified
   * as the listener for all menu items.  Note the use of getActionCommand()
   * to determine the command string registered with the individual items.
 ************************************************************************/
  public void actionPerformed(ActionEvent e) {
    String cmd = e.getActionCommand();
    if (cmd.equals("quit")) System.exit(0);   // Don't do this in an applet
    else if (cmd.equals("open")) load() ;     /*load from a file*/
    else if (cmd.equals("save")) save() ;     /*save to a file*/
    else if (cmd.equals("print")) print();    /*make a hard copy*/
    else if (cmd.equals("start")) startQSYS();    // defined below
    else if (cmd.equals("rmlast")) rmlast();
    else if (cmd.equals("rmall")) clear();
    else if (cmd.equals("clearscreen")) clearscreen();
    else if (cmd.equals("fresh")) freshScreen();
    else if (cmd.equals("about")) about(); /* not yet implemented */ ;
  }

private String equationx =new String(); //string of equation dx/dt=...
private String equationy =new String(); //string of equation dy/dt=...
/**************************************/
/** compEqn():assembly equations     **/
/**************************************/

protected void compEqn() {

  String s1= new String("dx/dt =");
      if(par_a00!= (float) 0){
      s1= s1+ par_a00;
      }
      if(par_a10!= (float) 0){
         if(par_a10>(float) 0)
       s1= s1+"+"+ par_a10 +" x";
        else
```

```
        s1= s1+par_a10 +" x";
    }
   if(par_a01!= (float) 0){
    if (par_a01>(float) 0)
     s1= s1+"+"+ par_a01 +" y";
     else
        s1= s1+ par_a01 +" y";
    }
   if(par_a20!= (float) 0){
     if(par_a20>(float) 0)
     s1= s1+"+"+ par_a20 +" x*x";
     else
         s1= s1+ par_a20 +" x*x";
    }
   if(par_a11!= (float) 0){
    if(par_a11> (float) 0)
     s1= s1+"+"+ par_a11 +" x*y";
    else
     s1= s1+par_a11 +" x*y";
    }
   if(par_a02!= (float) 0){
    if(par_a02> (float) 0)
     s1= s1+"+"+ par_a02 +" y*y";
    else
      s1= s1+  par_a02 +" y*y";
    }

  String s2= new String("dy/dt =");
    if(par_b00!= (float) 0){
    s2= s2+ par_a00;
    }
    if(par_b10!= (float) 0){
     if (par_b10> (float) 0)
     s2= s2+"+"+ par_b10 +" x";
```

```
        else
         s2= s2+par_b10 +" x";
       }
      if(par_b01!= (float) 0){
       if (par_b01 > (float) 0)
        s2= s2+"+"+ par_b01 +" y";
       else
         s2= s2+ par_b01 +" y";
       }
      if(par_b20!= (float) 0){
       if(par_b20> (float) 0)
        s2= s2+"+"+ par_b20 +" x*x";
        else
          s2= s2+ par_b20 +" x*x";
       }
      if(par_b11!= (float) 0){
       if(par_b11 > (float) 0)
        s2= s2+"+"+ par_b11 +" x*y";
       else
        s2= s2+ par_b11 +" x*y";
       }
      if(par_b02!= (float) 0){
       if (par_b02 > (float) 0)
        s2= s2+"+"+ par_b02 +" y*y";
       else
        s2= s2+ par_b02 +" y*y";
       }
 /**assign to the global variables**/
    equationx =s1;
    equationy =s2;
 }

/**start():initialize a phase drawing**/
protected void startQSYS() {
```

```
    clear();
    compEqn();
    InfoDialog d = new InfoDialog(frm, "Display Equations",
                        "You are visualizing the quadratic system:\n"+"\n"+
                        equationx +"\n" + equationy+"\n");
       d.setFont(font_small);
       d.show();
}

/*************************************/
/******load(): load from  a file ******/
protected void load() {
     float flt, flt1;
     int   tmpt , tmp_pointnum = 0;
     String  directory;
     String tmpstr = new String();
     char[] buffer = new char[258];
     BufferedReader in;
     FileDialog fi = new FileDialog(frm, "Load File", FileDialog.LOAD) ;
     File fl;

     pointnum=0;

    try {
      fi.show();                          //Display dialog and wait for response
      directory = fi.getDirectory();      //Remember new default directory

     fl = new File(directory, fi.getFile());
     in = new BufferedReader(new FileReader(fl));

    ////////////////////////////////
    /**** Read coefficients    ****/
      tmpstr = in.readLine();
```

```
if (!tmpstr.equals ("QDSDATA123.456")){
InfoDialog d = new InfoDialog(frm, "Wrong File",
                    "This is not a QDSSYS Data file!\n");
 d.show();
 return;
 }

tmpstr = in.readLine();
par_a00= (float) Float.valueOf( tmpstr).floatValue();
a_00.setText(tmpstr);
tmpstr = in.readLine();
par_a10= (float) Float.valueOf( tmpstr).floatValue();
a_10.setText(tmpstr);
tmpstr = in.readLine();
par_a01= (float) Float.valueOf( tmpstr).floatValue();
a_01.setText(tmpstr);
tmpstr = in.readLine();
par_a20= (float) Float.valueOf( tmpstr).floatValue();
a_20.setText(tmpstr);
tmpstr = in.readLine();
par_a11= (float) Float.valueOf( tmpstr).floatValue();
a_11.setText(tmpstr);
tmpstr = in.readLine();
par_a02= (float) Float.valueOf( tmpstr).floatValue();
a_02.setText(tmpstr);

tmpstr = in.readLine();
par_b00= (float) Float.valueOf( tmpstr).floatValue();
b_00.setText(tmpstr);
tmpstr = in.readLine();
par_b10= (float) Float.valueOf( tmpstr).floatValue();
b_10.setText(tmpstr);
tmpstr = in.readLine();
par_b01= (float) Float.valueOf( tmpstr).floatValue();
```

```
      b_01.setText(tmpstr);
      tmpstr = in.readLine();
      par_b20= (float) Float.valueOf( tmpstr).floatValue();
      b_20.setText(tmpstr);
      tmpstr = in.readLine();
      par_b11= (float) Float.valueOf( tmpstr).floatValue();
      b_11.setText(tmpstr);
      tmpstr = in.readLine();
      par_b02= (float) Float.valueOf( tmpstr).floatValue();
      b_02.setText(tmpstr);

    ////////////////////////////////////
    /****  Read control parameters ****/

    tmpstr = in.readLine();
    xmin = (float) Float.valueOf( tmpstr).floatValue() ;
    entry_xmin.setText(tmpstr);

    tmpstr = in.readLine();
    xmax = (float) Float.valueOf( tmpstr).floatValue() ;
    entry_xmax.setText(tmpstr);

    tmpstr = in.readLine();
    ymin = (float) Float.valueOf( tmpstr).floatValue() ;
    entry_ymin.setText(tmpstr);

    tmpstr = in.readLine();
    ymax = (float) Float.valueOf( tmpstr).floatValue() ;
    entry_ymax.setText(tmpstr);

    tmpstr = in.readLine();
    pointnum = Integer.valueOf(tmpstr).intValue();
////////////////////////////////////
/**** read all the initial data ****/
```

```
/****an orbit depends on initial****/
/**position,step size,step numberi**/
 for(int i=0; i <pointnum; i++){
     tmpstr = in.readLine();
     flt= (float) Float.valueOf( tmpstr).floatValue();
     pointx[i] = new Float(flt);

     tmpstr = in.readLine();
     flt= (float) Float.valueOf( tmpstr).floatValue();
     pointy[i] = new Float(flt);

    tmpstr = in.readLine();
    flt= (float) Float.valueOf( tmpstr).floatValue();
    cur_step[i] = new Float(flt);

    tmpstr = in.readLine();
    tmpt= (int) Integer.valueOf( tmpstr).intValue();
    cur_step_num[i] = new Integer(tmpt);
   }
 in.close();
  }
catch (IOException e) {
   System.err.println(e);}
   fi.dispose();

 Graphics gc = this.getGraphics();
 gc.setColor(this.getBackground());
 gc.clearRect(xshift, yshift, w , w );
 plotHead(gc);
 allPaint(gc);

    compEqn();
    InfoDialog d = new InfoDialog(frm, "Display Equations",
                          "You are visualizing the quadratic system:\n"+"\n"+
```

```
                        equationx +"\n" + equationy+"\n");
      d.setFont(font_small);
      d.show();
}

/*****************************/
/***save(): save to a file***/
protected void save() {
     float flt;
     int   tmpt, id = 0;
     String tmps ;
     char[] buffer = new char[258];
     String  directory;

     FileDialog fi = new FileDialog(frm, "Save File", FileDialog.SAVE) ;
     File fl;

    try {
     fi.show();                          // Display dialog and wait for response
     directory = fi.getDirectory();      // Remember new default directory

     fl = new File(directory, fi.getFile());
     FileWriter w = new FileWriter(fl);

    ////////////////////////
    /****save coeffients*****/

     tmps = String.valueOf("QDSDATA123.456");
     w.write(tmps.toCharArray()); w.write("\n");

     tmps = String.valueOf(par_a00);
     w.write(tmps.toCharArray()); w.write("\n");
     tmps = String.valueOf(par_a10);
     w.write(tmps.toCharArray()); w.write("\n");
```

```
tmps = String.valueOf(par_a01);
w.write(tmps.toCharArray()); w.write("\n");
tmps = String.valueOf(par_a20);
w.write(tmps.toCharArray());w.write("\n");
tmps = String.valueOf(par_a11);
w.write(tmps.toCharArray()); w.write("\n");
tmps = String.valueOf(par_a02);
w.write(tmps.toCharArray()); w.write("\n");

tmps = String.valueOf(par_b00);
w.write(tmps.toCharArray()); w.write("\n");
tmps = String.valueOf(par_b10);
w.write(tmps.toCharArray()); w.write("\n");
tmps = String.valueOf(par_b01);
w.write(tmps.toCharArray()); w.write("\n");
tmps = String.valueOf(par_b20);
w.write(tmps.toCharArray());w.write("\n");
tmps = String.valueOf(par_b11);
w.write(tmps.toCharArray()); w.write("\n");
tmps = String.valueOf(par_b02);
w.write(tmps.toCharArray()); w.write("\n");

//////////////////////////////
/***save control parameters***/
tmps = String.valueOf( xmin);
w.write(tmps.toCharArray()); w.write("\n");
tmps = String.valueOf( xmax);
w.write(tmps.toCharArray()); w.write("\n");
tmps = String.valueOf( ymin);
w.write(tmps.toCharArray()); w.write("\n");
tmps = String.valueOf( ymax);
w.write(tmps.toCharArray()); w.write("\n");
tmps = String.valueOf(pointnum);
w.write(tmps.toCharArray()); w.write("\n");
```

```
   ////////////////////////////
   /***save initial values***/
   for(int i=0; i <pointnum; i++){
     tmps = String.valueOf(pointx[i].floatValue());
     w.write(tmps.toCharArray()); w.write("\n");
     tmps = String.valueOf(pointy[i].floatValue());
     w.write(tmps.toCharArray());w.write("\n");
     tmps = String.valueOf(cur_step[i].floatValue());
     w.write(tmps.toCharArray());w.write("\n");
     tmps = String.valueOf(cur_step_num[i].intValue());
     w.write(tmps.toCharArray());w.write("\n");
    }
    w.flush();
    w.close();
   }
  catch (IOException e) {
  System.err.println(e) ;}
   fi.dispose();
 }

/*******************************/
/** print(): print a hard copy**/

protected void print() {

 int tmp_xprev, tmp_yprev;
 double  ttx, tty;

 /**set x-axis and y-axis**/
        to_xprev();
        to_yprev();

 PhaseWriter out = null;
```

```
 try { out = new PhaseWriter(frm, "Phase Portrait",10, 1.75, .75, 1.75, .75);}

 catch (PhaseWriter.PrintCanceledException e) { System.exit(0); }
 out.phasePlaneSetup("dx/dt=y", "dy/dt=-x");

// out.page.drawLine(0 + xshift, yorigin, w + xshift , yorigin);
// out.page.drawLine(xorigin, yshift , xorigin, w + yshift-1);

out.phaseDrawLine(0 + xshift, yorigin, w + xshift , yorigin);
out.phaseDrawLine(xorigin, yshift , xorigin, w + yshift-1);

 for(int j=0; j<pointnum ; j++){
        xprevf = (float) pointx[j].floatValue();
        yprevf = (float) pointy[j].floatValue();
        delta_t = (float) cur_step[j].floatValue();
        imax = (int) cur_step_num[j].intValue();

        to_xprev();
        to_yprev();

   for (int i = 1; i <=imax; i++) {
          tmp_xprev = xprev;
          tmp_yprev = yprev;
          Runge_Kutta();
          to_xprev();
          to_yprev();

      if (((xprev != tmp_xprev) || (yprev != tmp_yprev)) &&(yprev >yshift)
        &&(tmp_yprev>yshift) &&(xprev >xshift) && (tmp_xprev > xshift)&&
        (xprev <w +xshift) && (tmp_xprev <w +xshift))
// out.page.drawLine(tmp_xprev, tmp_yprev,  xprev , yprev);
out.phaseDrawLine(tmp_xprev, tmp_yprev,  xprev , yprev);
 }
}
```

```
  ////////////////////////
  /***design page head ***/
  String s= new String("Quadratic System");
  out.phaseDrawString(s, out.getUpperLeftX(), out.getUpperLeftY()+10);

  compEqn();
  out.phaseDrawString(equationx, out.getUpperLeftX(),out.getUpperLeftY()+30);

  out.phaseDrawString(equationy, out.getUpperLeftX(), out.getUpperLeftY()+40);

  String s3= new String("Range:  ");
  s3= s3 + "Min X=" + xmin + ",Max X=" +xmax +";Min Y="+ymin+",MaxY="+ymax;
   out.page.drawString(s3, out.getUpperLeftX(), out.getUpperLeftY() +60);
 out.close();
}

/*************************************/
/** clear(): clear internal data      **/
/**          Used by actionPerformed()**/
  protected void clear() {
    Graphics gc = this.getGraphics();
    gc.setColor(this.getBackground());
    gc.clearRect(xshift, yshift, w , w );
    colorID =0;
    pointnum = 0;
    plotHead(gc);
  }

/*****************************************/
/**remove last orbit :Display the remaining**/
 protected void rmlast() {

 Graphics gc = this.getGraphics();
```

```
 if (pointnum>0) {
  pointnum--;
 gc.setColor(this.getBackground());
 gc.clearRect(xshift, yshift, w , w );
 plotHead(gc);
 allPaint(gc);
 }
}

/************************************/
/**clearscreen(): Clear screen only **/
protected void clearscreen() {
    Graphics gc = this.getGraphics();
    gc.setColor(this.getBackground());
    gc.clearRect(xshift, yshift, w , w );
    plotHead(gc);
  }

/************************************/
/**freshScreen():Display all internal**/
 protected void freshScreen() {

 Graphics gc = this.getGraphics();

 gc.setColor(this.getBackground());
 gc.clearRect(xshift, yshift, w , w );
 plotHead(gc);
 allPaint(gc);
}

/*********************************/
/**about(): about this program***/
```

```
protected void about() {

InfoDialog d = new InfoDialog(frm, "About",
       "This Program  was written by Xianhua Huang\n" +
      "Copyright (c) 1998 Concordia University\n\n" +
      "It is to visualize  quadratic systems:   \n"+
      "dx/dt=a_00 +a_10 x +a_01 y + a_20 x*x+ a_11x*y +a_02 y*y\n" +
      "dy/dt=b_00 +b_10 x +b_01 y + b_20 x*x+ b_11x*y +b_02 y*y\n\n" +
      "\nQSYS (Under Java 1.1)\n"+
      "\n\n"+ "Usage:\n\n"+
      "Double Click : choose an initial point and draw a orbit\n"+
      "Menu Functions:\n"+
      "     File :  \n"+
      "          Start: clear internal data and get the program ready to work\n
      "          Load: load a previous saved file\n"+
      "          Save: save current status to a file\n"+
      "          Print: produce a hard copy\n"+
      "          Quit:  quit the program\n"+
      "     Edit : \n"+
      "           Delete Last Orbit: remove the most recent orbit\n"+
      "           Delete All: remove all the orbits and restart\n"+
      "     Screen :  \n"+
      "           Clear Screen: clear screen and keep the internal data\n"+
      "           Fresh Screen: display all the internal data\n"+
      "     Help: \n" +
      "           About: information about this program\n\n\n\n"+
     "Ranges:\n"+
     "   a_00, a_10,  a_01, a_20, a_11, a_02, \n"+
     "   b_00, b_10,  b_01, b_20, b_11, b_02 \n" +
     "   Min X, Max X, Min Y, Max Y\n" +
     "   are defined between -9999 and 9999 by QSYS\n"+ "\n" +
     "Constraint:\n"+
     "     Minimum X must be less than Maximum X; \n"+
     "     Minimum Y must be less than Maximum Y. \n"+
```

```
        "      Step Size must be between -.4 and .4\n"+
        "      Step# must be between 10 and 9999\n");
      d.show();
 }

/*********************************/
/******Interface Definition******/
/*********************************/
 protected PopupMenu popup_file;
 protected PopupMenu popup_edit;
 protected PopupMenu popup_screen;
 protected PopupMenu popup_help;

 FloatTextField a_00 =new FloatTextField("0.0", (float)-9999.,(float)9999, 2);
 FloatTextField a_10 =new FloatTextField("0.3", (float)-9999.,(float)9999, 3);
 FloatTextField a_01 =new FloatTextField("-0.1", (float)-9999.,(float)9999, 3);
 FloatTextField a_20 =new FloatTextField("-3.0", (float)-9999.,(float)9999, 3);
 FloatTextField a_11 =new FloatTextField("0.0", (float)-9999.,(float)9999, 3);
 FloatTextField a_02 =new FloatTextField("0.0", (float)-9999.,(float)9999, 3);

 FloatTextField b_00 = new FloatTextField("0.0",(float)-9999.,(float)9999, 2);
 FloatTextField b_10 = new FloatTextField("1.0",(float)-9999.,(float)9999, 3);
 FloatTextField b_01 = new FloatTextField("0.0",(float)-9999.,(float)9999, 3);
 FloatTextField b_20 = new FloatTextField("2.2",(float)-9999.,(float)9999, 3);
 FloatTextField b_11 = new FloatTextField("0.0",(float)-9999.,(float)9999, 3);
 FloatTextField b_02 = new FloatTextField("0.0",(float)-9999.,(float)9999, 3);

 FloatTextField entry_xmax=new FloatTextField("1.",(float)-9999,(float)9999,4);
 FloatTextField entry_xmin=new FloatTextField("-1.",(float)-9999,(float)9999,4);
 FloatTextField entry_ymax=new FloatTextField("1.",(float)-9999,(float)9999,4);
 FloatTextField entry_ymin=new FloatTextField("-1.",(float)-9999,(float)9999,4);
 FloatTextField entry_delta_t =new FloatTextField("0.02",(float)-.4,(float).4, 4
 FloatTextField entry_stepNum =new FloatTextField("1500",(float)10,(float)9999,
```

```
private Label x1;
private Label y1;
private Label xx;
private Label xy;
private Label yy;
private Label dx;
private Label plus;

private Label x1b;
private Label y1b;
private Label xxb;
private Label xyb;
private Label yyb;
private Label dy;
private Label dt;
private Label plusb;

//////////////////////
 /**   button        **/
private Button button_head;
private Button button_delete_one; // in use when it's applet

 /**      labels    **/
private Label lb_scale;
private Label lb_step;
private Label lb_stepsize;
private Label lb_xmax;
private Label lb_xmin;
private Label lb_ymax;
private Label lb_ymin;
private Panel row_head;
private Panel row1 ;
private Panel row2;
private Panel row3;
```

```
  private Panel row4;
  ////////////////////
  /**font definition**/
  private Font font;
  private Font font_small;
/*********************************/
/***end of interface definition***/
/*********************************/


/*********************************/
/***********Variables************/
/*********************************/

   /*****************************/
   /*******coefficients   ******/
    private float par_a00;
    private float par_a10;
    private float par_a01;
    private float par_a20;
    private float par_a11;
    private float par_a02;
    private float par_b00;
    private float par_b10;
    private float par_b01;
    private float par_b20;
    private float par_b11;
    private float par_b02;

/***********************/
/***internal variables**/
  private int yshift = 240;
  private int xshift = 70;
  private float xmax;
```

```
  private float xmin;
  private float ymax;
  private float ymin;
  private float delta_t;
  private int stepNum;
  private  int w = 450;
  private  int m = 1;
  private  int imax = 1500; // maximum steps of iteration
  private  int pointnum = 0;
  private  static Float pointx[] = new Float[201];
  private  static Float pointy[] = new Float[201];
  private  static Float cur_step[] = new Float[201];
  private  static Integer cur_step_num[] = new Integer[201];
  private float xn;
  private int xorigin=w/2 +xshift; //Screen Location of R^2 origin
  private int yorigin=w/2 +yshift;
  private int xprev=w/2+xshift;   //Initial point of an orbit in Screen
  private int yprev=w/2 +yshift;
  private float xprevf= (float) 0.0;//Initial point of an orbit in R^2
  private float yprevf= (float) 0.0;

  private int colorID = 0;
  private Color[] colors_name ={Color.blue, Color.cyan, Color.gray,
                  Color.green, Color.red, Color.yellow, Color.pink,
                   Color.lightGray, Color.magenta, Color.orange};
/*********************************/
/*******End of Variables*********/
/*********************************/


/*****************************/
/**init(): entry of applets**/
/*****************************/
 public void init() {
```

```
// If we are not in a frame (i.e. we are an applet), create a popup menu
   if (!(this.getParent() instanceof Frame)) {
// Create the popup menu
      popup_file = new PopupMenu("File");
      popup_edit = new PopupMenu("Edit");
      popup_screen = new PopupMenu("Screen");
      popup_help = new PopupMenu("Help");

 // Add items to it using the convenience routine below
      createMenuItems(popup_file, this,
         new String[] {"Start", "Load", "Save", "Print", "Quit"},
         new String[] {"clear","load", "save", "print", "quit"},
         new int[] {KeyEvent.VK_G,  KeyEvent.VK_L, KeyEvent.VK_S,
                    KeyEvent.VK_P, KeyEvent.VK_Q});

     createMenuItems(popup_edit, this,
         new String[] {"Delete Last Orbit", "Delete All"},
         new String[] {"rmlast", "rmall"},
         new int[] { KeyEvent.VK_R, KeyEvent.VK_M});

      createMenuItems(popup_screen, this,
         new String[] {"Clear Screen", "Fresh Screen"},
         new String[] {"clearscreen", "fresh"},
         new int[] { KeyEvent.VK_K, KeyEvent.VK_F});

      createMenuItems(popup_help, this,
         new String[] {"About"},
         new String[] { "about"},
         new int[] { KeyEvent.VK_A});


// Add the popup menu to the component it will appear over.
   this.add(popup_file);
```

```
    this.add(popup_edit);
    this.add(popup_screen);
    this.add(popup_help);
  }

/*************************/
/****design interface ****/
/*************************/
////////////////////
///Define head /////
    row_head = new Panel();
    row1     = new Panel();
    row2     = new Panel();
    row3     = new Panel();
    row4     = new Panel();

    row_head.setBackground(Color.blue);
    row_head.setForeground(Color.yellow);
        font = new Font("Helvetica", Font.BOLD, 26);
        font_small = new Font("Helvetica", Font.BOLD, 14);

        button_head = new Button("          Quadratic System:          ");
                button_head.setBackground(Color.yellow);
                button_head.setForeground(Color.blue);
                button_head.setFont(font);
                row_head.add(button_head);
        this.add(row_head, "Center");
//////end of head /////
//////define dx/dt/////
        dx =new Label("dx/dt=");
                dx.setBackground(Color.green);
                dx.setFont(font);
                dx.setForeground(Color.red);
                dx.setFont(font_small);
```

```
plus = new Label("+");
        plus.setBackground(Color.white);
        plus.setFont(font);
        plus.setForeground(Color.red);
        plus.setFont(font_small);
x1 = new Label("x+");
        x1.setBackground(Color.white);
        x1.setFont(font);
        x1.setForeground(Color.red);
        x1.setFont(font_small);
y1= new Label("y+");
        y1.setBackground(Color.white);
        y1.setFont(font);
        y1.setForeground(Color.red);
        y1.setFont(font_small);
 xx = new Label("xx+");
        xx.setBackground(Color.white);
        xx.setFont(font);
        xx.setForeground(Color.red);
        xx.setFont(font_small);
 xy = new Label("xy+");
        xy.setBackground(Color.white);
        xy.setFont(font);
        xy.setForeground(Color.red);
        xy.setFont(font_small);
 yy = new Label("yy");
        yy.setBackground(Color.white);
        yy.setFont(font);
        yy.setForeground(Color.red);
        yy.setFont(font_small);
        row1.add(dx);                    //dx/dt=
        row1.add(a_00);                  //a_00
        row1.add(plus);                  //+
        row1.add(a_10);                  //a_10
```

```
                row1.add(x1);                   //x +
                row1.add(a_01);                 //a_01
                row1.add(y1);                   //y +
                row1.add(a_20);                 //a_20
                row1.add(xx);                   //xx +
                row1.add(a_11);                 //a_11
                row1.add(xy);                   //xy +
                row1.add(a_02);                 //a_02
                row1.add(yy);                   //yy
                this.add(row1, "Center");       // bind row 1
//////end of dx/dt//////////

//////define dy/dt//////////
          dy =new Label("dy/dt=");
               dy.setBackground(Color.green);
               dy.setFont(font);
               dy.setForeground(Color.red);
               dy.setFont(font_small);
        plusb = new Label("+");
               plusb.setBackground(Color.white);
               plusb.setFont(font);
               plusb.setForeground(Color.red);
               plusb.setFont(font_small);
        x1b = new Label("x+");
               x1b.setBackground(Color.white);
               x1b.setFont(font);
               x1b.setForeground(Color.red);
               x1b.setFont(font_small);
        y1b= new Label("y+");
               y1b.setBackground(Color.white);
               y1b.setFont(font);
               y1b.setForeground(Color.red);
               y1b.setFont(font_small);
         xxb = new Label("xx+");
```

```
                xxb.setBackground(Color.white);
                xxb.setFont(font);
                xxb.setForeground(Color.red);
                xxb.setFont(font_small);
        xyb = new Label("xy+");
                xyb.setBackground(Color.white);
                xyb.setFont(font);
                xyb.setForeground(Color.red);
                xyb.setFont(font_small);
        yyb = new Label("yy");
                yyb.setBackground(Color.white);
                yyb.setFont(font);
                yyb.setForeground(Color.red);
                yyb.setFont(font_small);

                row2.add(dy);                   //dy/dt=
                row2.add(b_00);                 //b_00
                row2.add(plusb);                 // +
                row2.add(b_10);                 //b_10
                row2.add(x1b);                   //x +
                row2.add(b_01);                 //b_01
                row2.add(y1b);                   //y +
                row2.add(b_20);                 //b_20
                row2.add(xxb);                   //xx +
                row2.add(b_11);                 //b_11
                row2.add(xyb);                   //xy +
                row2.add(b_02);                 //b_02
                row2.add(yyb);                   //yy
                this.add(row2, "Center");       //bind row 2
///////end of dy/dt////////

///////define scales, step size, step number////////
        lb_scale = new Label("Ranges: ");               //Ranges:--
              lb_scale.setBackground(Color.green);
```

```
            lb_scale.setFont(font);
            lb_scale.setForeground(Color.red);
            lb_scale.setFont(font_small);
            row3.add(lb_scale);

        lb_xmin = new Label("Min X");                    //Min X
            lb_xmin.setBackground(Color.green);
            lb_xmin.setFont(font);
            lb_xmin.setForeground(Color.red);
            lb_xmin.setFont(font_small);
            row3.add(lb_xmin);
            row3.add(entry_xmin);

         lb_xmax = new Label("Max X");                    //Max X
            lb_xmax.setBackground(Color.green);
            lb_xmax.setFont(font);
            lb_xmax.setForeground(Color.red);
            lb_xmax.setFont(font_small);
            row3.add(lb_xmax);
            row3.add(entry_xmax);

         lb_ymin = new Label("Min Y");                    //Min Y
            lb_ymin.setBackground(Color.green);
            lb_ymin.setFont(font);
            lb_ymin.setForeground(Color.red);
            lb_ymin.setFont(font_small);
            row3.add(lb_ymin);
            row3.add(entry_ymin);

         lb_ymax = new Label("Max Y");                    //Max Y
            lb_ymax.setBackground(Color.green);
            lb_ymax.setFont(font);
            lb_ymax.setForeground(Color.red);
            lb_ymax.setFont(font_small);
```

```
            row3.add(lb_ymax);
            row3.add(entry_ymax);
        this.add(row3, "Center");

         lb_stepsize  = new Label("Step Size");              // dt or (delta t)
            lb_stepsize.setBackground(Color.green);
            lb_stepsize.setFont(font);
            lb_stepsize.setForeground(Color.red);
            lb_stepsize.setFont(font_small);
            row4.add(lb_stepsize);
            row4.add(entry_delta_t);

          lb_step = new Label("Step#");                    //Step Number:
            lb_step.setBackground(Color.green);
            lb_step.setFont(font_small);
            lb_step.setForeground(Color.red);
            lb_step.setFont(font_small);
            row4.add(lb_step);
            row4.add(entry_stepNum);

          /*Set up a "Delete Last OrbitI" button, when it is an applet*/
          if (!(this.getParent() instanceof Frame)) {
          button_delete_one = new Button("Delete Last Orbit");
          button_delete_one.setFont(font_small);
          button_delete_one.setBackground(Color.yellow);
          button_delete_one.setForeground(Color.red);
          row4.add(button_delete_one);
          }
          this.add(row4, "Center");
///////end od scale definition//////

/***End of interface design***/
/*****************************/
```

```
    resize(w ,w+yshift+10 );
    this.addMouseListener(this);
  }

/****End of init()****/
/*********************/


/**************************/
/****Graphics Functions****/
/**************************/

public void paint(Graphics g) {

plotHead(g);
allPaint(g);
}

/***********************************/
/***crespond to each double click***/
public void singlePaint(Graphics g) {

  int tmp_xprev, tmp_yprev;
  double  ttx, tty;

  g.setColor( colors_name[ colorID++ % 10] );

  for (int i = 1; i <=stepNum; i++) {
  tmp_xprev = xprev;
  tmp_yprev = yprev;
  Runge_Kutta();
  to_xprev();
  to_yprev();
```

```
  //To avoid duplicate drawing
  if (((xprev != tmp_xprev) || (yprev != tmp_yprev)) &&(yprev >yshift)
   &&(tmp_yprev>yshift)&&(yprev <yshift +w) &&(tmp_yprev<yshift+w)&&
   (xprev >xshift) && (tmp_xprev >xshift)&&(xprev <w+xshift) &&
   (tmp_xprev <w+xshift))
  g.drawLine(tmp_xprev, tmp_yprev,  xprev , yprev);
  }
 }


/*********************************/
/****       for update       ****/
public void allPaint(Graphics g) {

  int tmp_xprev, tmp_yprev;
  double  ttx, tty;

  plotHead(g);

  colorID = 0;

 for(int j=0; j<pointnum ; j++){
  g.setColor( colors_name[ colorID++ % 10] );

  xprevf = (float) pointx[j].floatValue();
  yprevf = (float) pointy[j].floatValue();
  delta_t = (float) cur_step[j].floatValue();
  imax = (int) cur_step_num[j].intValue();
 to_xprev();
 to_yprev();

  for (int i = 1; i <=imax; i++) {
  tmp_xprev = xprev;
  tmp_yprev = yprev;
```

```
  Runge_Kutta();
  to_xprev();
  to_yprev();

   //To avoid duplicate drawing
  if (((xprev != tmp_xprev) || (yprev != tmp_yprev)) &&(yprev >yshift)
   &&(tmp_yprev>yshift) &&(yprev <yshift+w) &&(tmp_yprev<yshift+w)&&
   (xprev >xshift) && (tmp_xprev >xshift)&&(xprev <w+xshift) &&
    (tmp_xprev <w +xshift))
  g.drawLine(tmp_xprev, tmp_yprev,  xprev , yprev);

  }
 }
}

/*********************************/
/*****       plot head     *****/
public void plotHead(Graphics g) {

     par_a00 = a_00.getValue();
     par_a10 = a_10.getValue();
     par_a01 = a_01.getValue();
     par_a20 = a_20.getValue();
     par_a11 = a_11.getValue();
     par_a02 = a_02.getValue();
     par_b00 = b_00.getValue();
     par_b10 = b_10.getValue();
     par_b01 = b_01.getValue();
     par_b20 = b_20.getValue();
     par_b11 = b_11.getValue();
     par_b02 = b_02.getValue();
     xmin    =entry_xmin.getValue();
     xmax    =entry_xmax.getValue();
     ymin    =entry_ymin.getValue();
```

```
      ymax    =entry_ymax.getValue();
      delta_t =entry_delta_t.getValue();
      stepNum = (int) entry_stepNum.getValue();
      imax = (int) stepNum;
 ///draw boundary//////
     g.setColor(Color.black);
     g.drawLine(0 + xshift,  yshift, w +xshift ,  yshift);
     g.drawLine(0 + xshift , w+yshift , w +xshift, w + yshift);
     g.drawLine(1 + xshift ,  yshift, 1+xshift , w + yshift);
     g.drawLine(w +xshift ,  yshift, w +xshift , w + yshift);
 /////draw x-axis and y-axis//////
     /**set x-axis and y-axis**/
         to_xprev();
         to_yprev();
     g.setColor(Color.green);
     g.drawLine(xshift, yorigin, w +xshift, yorigin);
     g.drawLine(xorigin, yshift, xorigin, w + yshift-1);

 }

 /*********************************/
 /****End of Graphics Functions****/
 /*********************************/

 /*****************************************/
 /**Transformations between R^2 and Screen**/
 /*****************************************/

  ////////////////////////////
  ////From R^2 to Screen//////
 private void to_xprev() {
 xprev = xshift + (int) ((float)  w* (xprevf- xmin)/(xmax - xmin));
 xorigin = xshift + (int) ((float)  w* (-xmin)/(xmax - xmin));
 }
```

```
private void to_yprev() {
yprev = yshift +w - (int)((float) w* (yprevf-ymin)/(ymax - ymin));
yorigin = yshift +w - (int)((float) w* (-ymin)/(ymax - ymin));
}

 ////////////////////////////
 ////From Screen to R^2//////
private void to_xprevf() {
xprevf = xmin +  (float) (xprev-xshift)*  (xmax - xmin)/(float) w;
}

private void to_yprevf() {
yprevf = ymin  - (float) (yprev - yshift -w) *(ymax- ymin)/ (float) w;
}



/*************************************/
/**Definition of Systems, Computation**/
/*************************************/
////////////////////////////////////
//////Definition of Systems/////////
public float dot_x(float x, float y) {
  return (par_a00 +par_a10*x +par_a01*y+par_a20* x*x
          +par_a11*x*y + par_a02 *y*y);
}

public float dot_y(float x, float y) {
 return (par_b00 +par_b10*x +par_b01*y +par_b20* x*x
         +par_b11*x*y + par_b02 *y*y);
}
```

```
////////////////////////////////////
//////////Runge_Kutta integral ///////
/*** 4th order Runge Kutta Method ***/
public void  Runge_Kutta() {

float tmp_x1;
float tmp_y1;
float tmp_dotx1, tmp_dotx2, tmp_dotx3, tmp_dotx4;
float tmp_doty1, tmp_doty2, tmp_doty3, tmp_doty4;
float ww;

tmp_dotx1 = dot_x(xprevf, yprevf);
tmp_doty1 = dot_y(xprevf, yprevf);

tmp_x1 = (float) xprevf + (float) delta_t*(float) tmp_dotx1/(float) 2.0;
tmp_y1 = (float) yprevf+  (float) delta_t*(float) tmp_doty1/(float) 2.0;
tmp_dotx2 =dot_x(tmp_x1, tmp_y1);
tmp_doty2 =dot_y(tmp_x1, tmp_y1);

tmp_x1= (float)xprevf + (float)delta_t*(float) tmp_dotx2/(float) 2.0;
tmp_y1= (float)yprevf+ (float) delta_t*(float) tmp_doty2/(float) 2.0;
tmp_dotx3 =dot_x(tmp_x1, tmp_y1);
tmp_doty3 =dot_y(tmp_x1, tmp_y1);

tmp_x1=  (float) xprevf + (float) delta_t*(float)tmp_dotx3;
tmp_y1=  (float) yprevf + (float) delta_t*(float)tmp_doty3;
tmp_dotx4 =dot_x(tmp_x1, tmp_y1);
tmp_doty4 =dot_y(tmp_x1, tmp_y1);

xprevf += (tmp_dotx1 +2.0*tmp_dotx2 +2.0* tmp_dotx3 +tmp_dotx4)* delta_t/6.0;
yprevf += (tmp_doty1 +2.0*tmp_doty2 +2.0* tmp_doty3 +tmp_doty4)* delta_t/6.0;
}

/*********************************************/
```

```
/**End Of Definition of Systems, Computation**/
/*********************************************/

/*****************************************/
/*Methods of the mouseListener interface*/
/*****************************************/
public void  mouseClicked(MouseEvent evt) {

 Graphics tmp_g;
 tmp_g = this.getGraphics();

 if (evt.getClickCount() == 2) {
     xprev =   evt.getX();
     yprev =   evt.getY();

  /* If click outside the graphics area, no action*/
  if ((xprev< xshift +w) && (xprev> xshift ) &&
       (yprev< yshift +w) && (yprev> yshift )) {
     to_xprevf();
     to_yprevf();

   pointx[pointnum] = new Float(xprevf);
   pointy[pointnum] = new Float(yprevf);
   delta_t = entry_delta_t.getValue();
   stepNum = (int) entry_stepNum.getValue();
   cur_step[pointnum] = new Float(delta_t);
   cur_step_num[pointnum] = new Integer(stepNum);
   pointnum++;

if (all_valid()){
 singlePaint(tmp_g); }
 else
    pointnum--;
   }
```

```
  }
}

public void  mousePressed(MouseEvent evt) { ; }

public void  mouseReleased(MouseEvent evt) { ; }

public void  mouseExited(MouseEvent evt) {;}

public void  mouseEntered(MouseEvent evt) {;}

public void  mouseDragged(MouseEvent evt) {;}

/************************************/
/*****     Check entries       *****/
/************************************/
private  boolean all_valid() {
 InfoDialog d;
 float  tmp_xmin, tmp_xmax, tmp_ymin, tmp_ymax;
   tmp_xmin= xmin;
   tmp_xmax= xmax;
   tmp_ymin= ymin;
   tmp_ymax= ymax;

 Graphics gc = this.getGraphics();
  plotHead(gc);

  if((tmp_xmin!= xmin)||(tmp_xmax!= xmax)||(tmp_ymin!= ymin)||(tmp_ymax!=ymax))
   allPaint(gc);

  if(!(entry_ymax.isValid()&& entry_ymin.isValid()
      &&entry_xmax.isValid()&&entry_xmin.isValid()))
  {  d = new InfoDialog(frm, "Hints",
      "Note:" + "\n"+
```

```
      " One of Min X , Max X ,Min Y, Max Y in invalid.\n"+
      " They must be float and between -9999 and 9999\n");
       d.setFont(font_small);
       d.show();
    return false;
  }

  if ((xmin >= xmax) || (ymin >= ymax) ){
      d = new InfoDialog(frm, "Hints",
      "Note:" + "\n"+
      " Min X must be less than Max X \n"+
      " Min Y must be less than Max Y\n");
      d.setFont(font_small);
      d.show();
      return false;
     }

   if(!entry_delta_t.isValid()) {
      d = new InfoDialog(frm, "Hints",
      "Note:" + "\n"+
      " Entry Step Size is invalid: Wrong data type or too big!\n"+
      " It must be a float number\n");
      d.setFont(font_small);
      d.show();
      return false;
}

  if ((delta_t<.00000001)&& (delta_t >-.00000001)) {
      d = new InfoDialog(frm, "Hints",
      "Note:" + "\n"+
      " Step Size is too small.\n");
      d.setFont(font_small);
      d.show();
      return false;
```

```
    }

   if(!entry_stepNum.isValid()) {
     d = new InfoDialog(frm, "Hints",
       "Note:" + "\n"+
       " Step#  is invalid: wrong data type or too large\n"+
       " It must be an integer number between 10 and 9999\n");
       d.setFont(font_small);
       d.show();
       return false;
       }

    if(stepNum<10) {
      d = new InfoDialog(frm, "Hints",
       "Note:" + "\n"+
       " Step#  must be between 10 and 9999\n");
       d.setFont(font_small);
       d.show();
       return false;
       }

 if (a_00.isValid()&& a_10.isValid()&&a_01.isValid()&&a_20.isValid()&&
     a_11.isValid()&& a_02.isValid()&& b_00.isValid()&& b_10.isValid()&&
     b_01.isValid()&& b_20.isValid()&& b_11.isValid()&& b_02.isValid())
 return true;
else {
  d = new InfoDialog(frm, "Hints",
       "Note:" + "\n"+
       " Coefficients are invalid: wrong data type or too big\n"+
       " They must be float numbers between -9999 and 9999\n");
       d.show();
       return false;
       }
}
```

```
/*********************************************/
/*End of Methods of the mouseListener interface*/
/*********************************************/

/*********************************************/
/***        Method used by applet          ***/
/*********************************************/
public boolean action(Event e, Object arg) {
        Object target = e.target;
          if (target == button_head) {
           clear();
         }
        else if (target == button_delete_one)
           {
           rmlast();
           }
          return true;
  }

}
```

B.2 PhaseWriter.java

```
/*****************************************************************
 ** PhaseWriter.java                                            **
 ** Produce a hard copy of phase portraits                      **
 ** Modified from HardcopyWriter.java (http://www.oreilly.com   **
 ** Nov 5, 1998                                                 **
 *****************************************************************/

/****************
 Original Notice:
```

```
 ****************/
// This example is from _Java Examples in a Nutshell_. (http://www.oreilly.com)
// Copyright (c) 1997 by David Flanagan
// This example is provided WITHOUT ANY WARRANTY either expressed or implied.
// You may study, use, modify, and distribute it for non-commercial purposes.
// For any commercial use, see http://www.davidflanagan.com/javaexamples

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.text.*;
import java.util.*;

/**  A character output stream that sends output to a printer.  **/
public class PhaseWriter extends Writer {
  // These are the instance variables for the class
  protected PrintJob job;                    // The PrintJob object in use
  protected Graphics page;                   // Graphics object for current page
  protected String jobname;                  // The name of the print job
  protected int fontsize;                    // Point size of the font
  protected String time;                     // Current time (appears in header)
  protected Dimension pagesize;              // Size of the page (in dots)
  protected int pagedpi;                     // Page resolution in dots per inch
  protected Font font, headerfont;           // Body font and header font
  protected FontMetrics metrics;             // Metrics for the body font
  protected FontMetrics headermetrics;       // Metrics for the header font
  protected int x0, y0;                      // Upper-left corner inside margin
  protected int width, height;               // Size (in dots) inside margins
  protected int headery;                     // Baseline of the page header
  protected int charwidth;                   // The width of each character
  protected int lineheight;                  // The height of each line
  protected int lineascent;                  // Offset of font baseline
  protected int chars_per_line;              // Number of characters per line
  protected int lines_per_page;              // Number of lines per page
```

```
protected int charnum = 0, linenum = 0;   // Current column and line position
protected int pagenum = 0;                // Current page number
public  Graphics phase;
// A field to save state between invocations of the write() method
private boolean last_char_was_return = false;

// A static variable that holds user preferences between print jobs
protected static Properties printprops = new Properties();

/**
 * The constructor for this class has a bunch of arguments:
 * The frame argument is required for all printing in Java.
 * The jobname appears left justified at the top of each printed page.
 * The font size is specified in points, as on-screen font sizes are.
 * The margins are specified in inches (or fractions of inches).
 **/
public PhaseWriter(Frame frame, String jobname, int fontsize,
                   double leftmargin, double rightmargin,
                   double topmargin, double bottommargin)
    throws PhaseWriter.PrintCanceledException
{
  // Get the PrintJob object with which we'll do all the printing.
  // The call is synchronized on the static printprops object, which
  // means that only one print dialog can be popped up at a time.
  // If the user clicks Cancel in the print dialog, throw an exception.
  Toolkit toolkit = frame.getToolkit();   // get Toolkit from Frame
  synchronized(printprops) {
    job = toolkit.getPrintJob(frame, jobname, printprops);
  }
  if (job == null)
    throw new PrintCanceledException("User cancelled print request");

  pagesize = job.getPageDimension();      // query the page size
  pagedpi = job.getPageResolution();      // query the page resolution
```

```
// Bug Workaround:
// On windows, getPageDimension() and getPageResolution don't work, so
// we've got to fake them.
if (System.getProperty("os.name").regionMatches(true,0,"windows",0,7)) {
  // Use screen dpi, which is what the PrintJob tries to emulate, anyway
  pagedpi = toolkit.getScreenResolution();
  System.out.println(pagedpi);
  // Assume a 8.5" x 11" page size.  A4 paper users have to change this.
  pagesize = new Dimension((int)(8.5 * pagedpi), 11*pagedpi);
  System.out.println(pagesize);
  // We also have to adjust the fontsize.  It is specified in points,
  // (1 point = 1/72 of an inch) but Windows measures it in pixels.
  fontsize = fontsize * pagedpi / 72;
  System.out.println(fontsize);
  System.out.flush();
}

// Compute coordinates of the upper-left corner of the page.
// I.e. the coordinates of (leftmargin, topmargin).  Also compute
// the width and height inside of the margins.
x0 = (int)(leftmargin * pagedpi);
y0 = (int)(topmargin * pagedpi);
width = pagesize.width - (int)((leftmargin + rightmargin) * pagedpi);
height = pagesize.height - (int)((topmargin + bottommargin) * pagedpi);

// Get body font and font size
font = new Font("Monospaced", Font.PLAIN, fontsize);
metrics = toolkit.getFontMetrics(font);
lineheight = metrics.getHeight();
lineascent = metrics.getAscent();
charwidth = metrics.charWidth('0');  // Assumes a monospaced font!

// Now compute columns and lines will fit inside the margins
```

```
    chars_per_line = width / charwidth;
    lines_per_page = height / lineheight;

    // Get header font information
    // And compute baseline of page header: 1/8" above the top margin
    headerfont = new Font("SansSerif", Font.ITALIC, fontsize);
    headermetrics = toolkit.getFontMetrics(headerfont);
    headery = y0 - (int)(0.125 * pagedpi) -
      headermetrics.getHeight() + headermetrics.getAscent();

    // Compute the date/time string to display in the page header
    DateFormat df = DateFormat.getDateTimeInstance(DateFormat.LONG,
                                                   DateFormat.SHORT);
    df.setTimeZone(TimeZone.getDefault());
    time = df.format(new Date());

    this.jobname = jobname;                 // save name
    this.fontsize = fontsize;               // save font size
  }

  /**
   * This is the write() method of the stream.  All Writer subclasses
   * implement this.  All other versions of write() are variants of this one
   **/
  public void write(char[] buffer, int index, int len) {
    synchronized(this.lock) {
      // Loop through all the characters passed to us
      for(int i = index; i < index + len; i++) {
        // If we haven't begun a page (or a new page), do that now.
        if (page == null) newpage();

        // If the character is a line terminator, then begin new line,
        // unless it is a \n immediately after a \r.
        if (buffer[i] == '\n') {
```

```
    if (!last_char_was_return) newline();
    continue;
  }
  if (buffer[i] == '\r') {
    newline();
    last_char_was_return = true;
    continue;
  }
  else last_char_was_return = false;

  // If it some other non-printing character, ignore it.
  if (Character.isWhitespace(buffer[i]) &&
      !Character.isSpaceChar(buffer[i]) && (buffer[i] != '\t')) continue;

  // If no more characters will fit on the line, start a new line.
  if (charnum >= chars_per_line) {
    newline();
    if (page == null) newpage();  // and start a new page, if necessary
  }

  // Now print the character:
  // If it is a space, skip one space, without output.
  // If it is a tab, skip the necessary number of spaces.
  // Otherwise, print the character.
  // It is inefficient to draw only one character at a time, but
  // because our FontMetrics don't match up exactly to what the
  // printer uses we need to position each character individually.
  if (Character.isSpaceChar(buffer[i])) charnum++;
  else if (buffer[i] == '\t') charnum += 8 - (charnum % 8);
  else {
    page.drawChars(buffer, i, 1,
                   x0 + charnum * charwidth,
                   y0 + (linenum*lineheight) + lineascent);
    charnum++;
```

```
        }
      }
    }
  }


/** This  method is  to set-up a phase plane. */
public void phasePlaneSetup(String str1, String str2) {
    page = job.getGraphics();                // Begin the new page
    phase = page;
    linenum = 0; charnum = 0;                // Reset line and char number
    pagenum++;                               // Increment page number
    page.setFont(headerfont);                // Set the header font.
  }

/**This  method is to draw a line on the phase portrait**/
public void phaseDrawLine(int x1, int y1, int x2, int y2) {
page.drawLine( x1, y1, x2, y2);
}


/**This  method is to draw a string on a designed location**/
public void phaseDrawString(String str, int x, int y) {
page.drawString(str, x, y);
}

public void phaseDrawOval(int point_x, int point_y, int r) {

page.drawOval(point_x, point_y, r, r);

  }

/**Get upper left coordinates of the phase plane**/
public int getUpperLeftX(){
```

```
 return x0;
}

public int getUpperLeftY(){
 return y0;
}

  /**
   * This is the flush() method that all Writer subclasses must implement.
   * There is no way to flush a PrintJob without prematurely printing the
   * page, so we don't do anything.
   **/
  public void flush() { /* do nothing */ }

  /**
   * This is the close() method that all Writer subclasses must implement.
   * Print the pending page (if any) and terminate the PrintJob.
   */
  public void close() {
    synchronized(this.lock) {
      if (page != null) page.dispose();   // Send page to the printer
      job.end();                          // Terminate the job
    }
  }

  /**
   * Set the font style.  The argument should be one of the font style
   * constants defined by the java.awt.Font class.  All subsequent output
   * will be in that style.  This method relies on all styles of the
   * Monospaced font having the same metrics.
   **/
  public void setFontStyle(int style) {
    synchronized (this.lock) {
      // Try to set a new font, but restore current one if it fails
```

```
    Font current = font;
    try { font = new Font("Monospaced", style, fontsize); }
    catch (Exception e) { font = current; }
    // If a page is pending, set the new font.  Otherwise newpage() will.
    if (page != null) page.setFont(font);
  }
}

/** End the current page.  Subsequent output will be on a new page. */
public void pageBreak() { synchronized(this.lock) { newpage(); } }

/** Return the number of columns of characters that fit on the page */
public int getCharactersPerLine() { return this.chars_per_line; }

/** Return the number of lines that fit on a page */
public int getLinesPerPage() { return this.lines_per_page; }

/** This internal method begins a new line */
protected void newline() {
  charnum = 0;                      // Reset character number to 0
  linenum++;                        // Increment line number
  if (linenum >= lines_per_page) {  // If we've reached the end of the page
    page.dispose();                 //    send page to printer
    page = null;                    //    but don't start a new page yet.
  }
}

/** This internal method begins a new page and prints the header. */
protected void newpage() {
  page = job.getGraphics();               // Begin the new page
  linenum = 0; charnum = 0;               // Reset line and char number
  pagenum++;                              // Increment page number
  page.setFont(headerfont);               // Set the header font.
  page.drawString(jobname, x0, headery);  // Print job name left justified
```

```
    String s = "- " + pagenum + " -";        // Print the page number centered.
    int w = headermetrics.stringWidth(s);
    page.drawString(s, x0 + (this.width - w)/2, headery);
    w = headermetrics.stringWidth(time);     // Print date right justified
    page.drawString(time, x0 + width - w, headery);

    // Draw a line beneath the header
    int y = headery + headermetrics.getDescent() + 1;
    page.drawLine(x0, y, x0+width, y);

    // Set the basic monospaced font for the rest of the page.
    page.setFont(font);
  }

  /**
   * This is the exception class that the HardcopyWriter constructor
   * throws when the user clicks "Cancel" in the print dialog box.
   **/
  public static class PrintCanceledException extends Exception {
    public PrintCanceledException(String msg) { super(msg); }
  }

}
```

Bibliography

[1] ARONSON, D.G., DOEDEL, E.J., OTHMER, H.G., *The dynamics of coupled current-biased Josephson junctions – Part II*, International Journal of Bifurcation and Chaos, **Vol. 1**, No. 1 (1991) 51-66

[2] BAMÓN, R., *Quadratic vector fields in the plane have a finite number of limit cycles*, Inst. Hautes. Etudes Sci. Publ. Math. **64** (1986), 111-142.

[3] BAUTIN,N.N., *On the number of limit cycles which appear with the variation of coefficients from an equilibrium position of focus or centre type*. Mat. Sb. **30(72)**(1952). 181-196 (Russian). Amer. Math Soc. Transl., **100**(1954). 396-413.

[4] BOGDANOV , R.I., *Versal deformation of a singularity of a vector field on the plane in the case of zero eigenvalues*, Select Math. Sov. **Vol. 1**, No. 4 (1981). 389-421.

[5] CAI, SUILIN AND WANG, ZHONGWEI, *A quadratic system with second order fine focus*. Chin. Ann. Mat. Ser.**Vol. 6A**, No. 5 (1984), 765-770. (Chinese)

[6] CHEN, LANSUN AND WANG, MINGSHU, *Relative position and number of limit cycles of a quadratic differential system*, Acta Math. Sinicai, **Vol. 22** (1979), 751-758. (Chinese)

[7] CHERKAS, L.A., *On the complex cycles of a certain differential equation*. Differentsial'nye Uravneniya **Vol.1**. No. 2 (1965), 182-186; *see also* Differential Equations. **Vol.1**. No. 2 (1965), 136-138.

[8] COLL, B., GASULL,A. AND LLIBRE, J., *Some theorems on the existence, uniqueness and non-existence of limit cycles for quadratic systems*, J. of Diff. Eqs., **67, No. 3**(1987), 372-399.

[9] COPPEL, W.A., *A survey of quadratic systems*. J. of Diff. Eqs., **2**(1966), 293-304.

[10] COPPEL, W.A., *Quadratic systems with a degenerate critical point*. Bull. Austral. Math. Soc., **Vol. 38** (1988), 1-10.

[11] DOEDEL, E., *Nonlinear numerics*. J. Franklin Inst. 1997

[12] DOEDEL, E., KELLER, H.B., KERNEVEZ, J.P., *Numerical analysis and control of bifurcation problems (I)*. International Journal of Bifurcation and chaos. **Vol.1**, No.3 (1991), 493-520

[13] DOEDEL, E., KELLER, H.B., KERNEVEZ, J.P., *Numerical analysis and control of bifurcation problems (II)*. International Journal of Bifurcation and chaos, **Vol.1**, No.3 (1991), 745-772

[14] DOEDEL, E., A.R. CHAMPNEYS ET AL, *AUTO97 : Continuation and bifurcation software for ordinary differential equations(with HomCont), User's Guide*, Concordia University, Montreal, Canada 1997.

[15] DUMORTIER, F. ET AL, *Bifurcations of Planar Vector Fields*. Springer-Verlag (1991)

[16] FULTON, W., *Algebraic curves, an introduction to algebraic geometry*, New York. Benjamin, 1969

[17] GOLUBITSKY, M., SCHAEFFER, D.G., *"Singularities and Groups in Bifurcation Theorey. Vol. I"*. Applied Math Science. **Vol. 51**, Springer-Verlag, New York. 1985.

[18] GOLUBITSKY, M., STEWART, I. AND SCHAEFFER, D.G., *"Singularities and Groups in Bifurcation Theorey. Vol. II"*, Applied Math Science. **Vol. 69**. Springer-Verlag, New York. 1988.

[19] GRIEWANK, A., REDDIEN, G.W., *Computation of cusp singularities for operator equations and their discretizations*, J. Comput. Appl. Maths. **26** (1989), 133-153)

[20] GUCKENHEIMER, J. HOLMES, P., *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*, Applied Math Science. **Vol. 42**, Springer-Verlag, New York, 1983

[21] HAN, MAOAN. *Uniqueness of limit cycles of a quadratic system of type $(III)_{n=0}$ around a focus of order 2.* Chin. Ann. Math. Ser. **A6**, No. 6 (1985), 661-668. (Chinese)

[22] HASSARD, B., HASTINGS, S., TROY, W. AND ZHANG, J., *A computer proof that the Lorenz equations have "chaotic" solutions*, Appl Math Letter.(to appear)

[23] HASTINGS, S., AND TROY, W. , *A shooting approach to the Lorenz equations*, Bulletin of the AMS, **27**(1992), 298-303

[24] HILBERT,D., *Mathematische Probleme*, Lecture, Second Internat. Congr. Math. (Paris, 1900), Nachr. Ges. Wiss. Göttingen Math,-Phys. Kl. 1900, 253-297; reprinted in Arch. Math. Phys. **1**(3) (1901), 44-63, 213-237 and also in his *Gesammelte Abhandlungen.* Vol. III, 2nd ed., Springer-Verlag, 1970, pp. 290-329; English trans. Bulletin Amer. Math. Soc. **8** (1902), 437-479; reprinted in Mathematical Developments Arising from Hilbert Problems, Proc. Symp. Pure Math., **Vol. 28**, Amer. Math. Soc., Providence, R.I., 1976, pp. 1-34.

[25] HUANG, X., REYN, J.W., *On the limit cycle distribution over two nests in quadratic systems*, Bull. of Austral. Math. Soc., **Vol. 52** (1995), 461-474.

[26] HUANG, X., REYN, J.W., *Bifurcations of nilpotent singularities and separatrix cycles in quadratic systems.* Proceedings of Dynamic Systems and Applications, **Vol. 2**, Dynamic Publishers, Atlanta, USA. (1996).

[27] HUANG, X., REYN, J.W., *Weak critical points and limit cycles in quadratic systems with finite multiplicity three.* Differential Equations and Dynamical Systems, **Vol. 5**, 3&4 (1997), 243-266.

[28] IL'YASHENKO, YU. S., *Finite Theorem for Limit Cycles*, Translation Mathematics Monographics **94**, AMS Providence (1991).

[29] JOYAL.P., ROUSSEAU.C., *Saddle quantities and applications*, J. of Diff. Eqs., **78**(1989), 374-399.

[30] KEARFORTT, R.B., *An interval step control for continuation methods*, SIAM J. Numerical Analysis, **31**(4)(1994), 892-914

[31] KELLER, H.B., *Numerical methods in bifurcation problems*. Springer-Verlag, 1987.

[32] KUZNETSOV, YU. A., *Elements of Applied Bifurcation Theory*. Applied Mathematics Sciences 112. Springer-Verlag, 1995

[33] LI, CHENGZHI, *Non-existence of limit cycles around a weak focus of order three for any quadratic system*. (Chinese summary appears in Chin. Ann. Math. Ser. **A7**, No. 2 (1986), 239.

[34] LI, CHENGZHI, LLIBRE,L.J. AND ZHANG, ZHIFEN, *Weak focus, limit cycles and bifurcations for bounded quadratic systems*, J. of Diff. Eqs., **Vol. 115**, No. 1 (1995), 193-223.

[35] LI, J., HUANG, Q. *Hilbert 16th problem when $n = 3$: $H(3) \geq 11$)*, Bulletin in Science **12**(1985), 958. (Chinese)

[36] LLOYD, N.G., *Small amplitude limit cycles of polynomial differential equations*, Ord. Diff. Eqs. and Oper., Lecture Notes in Math. **1032** (1982), 346-357

[37] MISCHAIKOW, K. AND MROZEK, M., *Chaos in the Lorenz equations: A computer assisted proof. Part II: Details*. Math. Comp. **67**(1998), 1023-1046

[38] PERKO,L.M., *Global families of limit cycles of planar analytic systems*. Trans. AMS, **Vol. 322**, No.2 (1990), 627-655

[39] PERKO,L.M., *Multiple limit cycle bifurcation surfaces and global families of multiple limit cycles*. **122**(98)(1995), 89-113

[40] PERKO, L.M., *Coppel's problem for bounded quadratic systems*. Dep. of Math. Northern Arizona University, Flagstaff, (August, 1994), 218 pp.

[41] PETROVSKII, I.G. AND LANDIS, E.M. *On the numer of limit cycles of the equation $dy/dx = P(x,y)/Q(x,y)$, where P and Q are polynomials of 2nd degree*. Mat. Sbornik (N.S.) **37** (79)(1955), 209-250, translation AMS Translations, Series 2, **Vol. 10** (1958), 177-221.

[42] REYN, J.W. *A bibliography of the qualitative theory of quadratic systems of differential equations in the plane*. Reports of the Faculty of Mathematics and Informatics, 94-02. Delft University of Technology, 3rd edition (1994)

[43] REYN, J.W., *Phase portraits of quadratic systems without finite critical points*, J. of Nonlinear Analysis. Theory, Methods and Applications. (to appear).

[44] REYN, J.W., *Phase portraits of non degenerate quadratic systems with finite multiplicity one*, J. of Nonlinear Analysis. Theory, Methods and Applications, (to appear).

[45] REYN, J.W., *Classes of quadratic systems of differential equations in the plane*. Proceedings of the Special Program at the Nankai Institute of Mathematics. Tianjin, P.R. China (September 1990-June 1991), Nankai Series in Pure, Applied. Mathematical and Theoretical Physics, **Vol.4**, Dynamical Systems, (1993),146-180

[46] REYN, J.W. AND HUANG, X., *Phase portraits of quadratic systems with finite multiplicity three and a degenerate critical point at infinity*, Rocky Mountain Journal, **27**(1998), 929.

[47] REYN, J.W. AND HUANG, X., *Phase portraits of quadratic systems with finite multiplicity three*, Proceedings of Dynamic Systems and Applications, **Vol. 2**, Dynamic Publishers. Atlanta. USA. (1996).

[48] REYN, J.W. AND KOOIJ, R.E., *Infinite singular points of quadratic systems in the plane*, J. of Nonlinear Analysis: Theory, Methods and Applications. **24**(6)(1995), 895-927.

[49] REYN, J.W. AND KOOIJ, R.E., *Phase portraits of non-degenerate quadratic systems with finite multiplicity two*. Differential Equations and Dynamical Systems. **Vol. 5**, 3&4 (1997). 355-414.

[50] ROOSE, D., PIESSENS, R., *Numerical computation of nonsimple turning points and cusps*, Numerische Mathematik. **46**(1985), 189-211.

[51] RYČKOV, G.S., *Some criteria for the presence and absence of limit cycles in a second order dynamical system*. Sibirsk. Math. Zh. **7**(1966), 1425-1431. (Russian)

[52] YE, YANQIAN AND OTHERS. *Theory of limit cycles*, Translations of Mathematical Monographs. **Vol.66**. Amer. Math. Soc., Providence, Rhode Island, U.S.A., 435 pp.

[53] SHI, SONGLING. *A concrete example of the existence of four limit cycles for plane quadratic systems*. Scientia Sinica. (Ser. A) **Vol.23**, No.2. (1980). 153-158.

[54] WANNER, G., *Über Shi's Gegenbeispiel zum 16. Hilbert problem*. Jahrbuch Überblicke Mathematik (1983). 9-24. Mannheim. (German)

[55] ZHANG, PINGGUANG, *On the concentrative distribution and uniqueness of limit cycles for a quadratic system*. Proc. of the Special Program at the Nankai Institute of Mathematics, Tianjin. PR. China. Nankai Dynamical Systems. (1993). 297-310.

[56] ZHANG, PINGGUANG AND CAI, SUILIN, *Quadratic systems with a weak focus*, Bull. Austral. Math. Soc., **44** (1991). 511-526.

[57] ZHANG, ZHI-FEN, DING, TONGREN, ET AL, *Qualitative theory of ordinary differential equations*, Translations of Mathematical Monographs, **Vol. 101**, Amer. Math. Soc., Providence, Rhode Island, U.S.A. (1992).

[58] ZHU, DEMING. *A general property of the quadratic differential systems*. Chin. Ann. of Math., **10B**(1)(1989). 26-32.