A Hybrid Symbolic-Numerical Approach in Computer Aided Geometric Design (CAGD) and Visualization

Dissertation

zur Erlangung des akademischen Grades
"Doktor der technischen Wissenschaften"
(Doctor of Technical Sciences)

eingereicht von

Trần Quốc–Nam

October 10, 1996

Erster Begutachter: Univ.-Doz. Dr. Franz Winkler
Zweiter Begutachter: Univ.-Doz. Dr. Andreas Neubauer

Angefertigt am
Research Institute for Symbolic Computation (RISC-Linz)
Technisch-Naturwissenschaftliche Fakultät
Johannes Kepler Universität Linz
Austria

Dedicated to my father **Trần Quốc–Diễm**

Eidesstattliche Erklärung

Ich versichere, daß ich die Dissertation selbständig verfaßt habe, andere als die angegebenen Quellen und Hilfsmittel nicht verwendet und mich auch sonst keiner unerlaubten Hilfe bedient habe.

Trần Quốc–Nam
Hagenberg, am 9. Oktober 1996.

Acknowledgments

Heartfelt thanks go to my thesis advisor, Univ.-Doz., Ass-Prof. Franz Winkler, for his support and encouragements throughout my studies and research.

I thank Univ.-Doz., Ass-Prof. Andreas Neubauer for serving on my thesis committee.

I thank Prof. Bruno Buchberger for his selfless dedication to the Research Institute for Symbolic Computation (RISC–Linz). He has created a fascinating environment for doing research.

My special thank goes to Prof. Gerard Huet, INRIA, France, for teaching me the theory of Term Rewriting and for introducing me to RISC–Linz.

No words can express my gratitude to my late father, my wife and my daughters, without their help and their patience this work could not be finished.

Several members of RISC–Linz contributed to this thesis in one way or another. In particular, I would like to thank Karel Stokkermans, Carl Van Geem and Emil Volcheck for their helps.

This work has been supported by the Austrian Science Foundation (FWF), project PoSSo and HySaX.

Abstract

Computer Aided Geometric Design (CAGD) and Visualization play an important role in the construction of such varied products as cars, ships, airplanes etc. A practical implementation has to balance two conflicting goals, namely efficiency and reliability. Typically, a numerical algorithm is efficient, but it is not reliable and may fail in certain cases, especially in areas near the singularities. On the other hand, algorithms based on exact arithmetic are reliable, but are normally slow and require a lot of memory space. Perhaps the goals of efficiency and reliability cannot be met simultaneously without some compromises.

In this dissertation, we try to negotiate those compromises judiciously. We have developed several hybrid symbolic-numerical methods as the basically mathematical methods to achieve our goals. Among them are: (a) Extended Newton's method. This is a new method for iteratively determining the roots of an arbitrary system of equations, where the number of equations of the system can be greater than, less than or equal to the number of variables. (b) Pseudoirreducible decomposition. The goal is to avoid expensive computations of irreducible decompositions, which is widely used in Algebraic Geometry, Computer Aided Geometric Design, etc., such as in parameterization, genus computation, projection of an algebraic set onto a hypersurface. We propose a new concept of pseudoirreducible decomposition which is an approximation of the irreducible decomposition and easy to compute. It turns out that this new decomposition is very useful in Computer Aided Geometric Design (CAGD) and Visualization, in particular for one-dimensional algebraic sets. (c) Extended Dixon's resultant. By using techniques from pseudoinverse theory, we extend Dixon's resultant for simultaneously eliminating several variables from a system of $n+1$ polynomials in n variables, where the Dixon matrix can be singular or even nonsquare. (d) An improved algorithm for real root isolation of polynomials with real algebraic number coefficients. By avoiding the use of unit transformation and utilizing interval arithmetic, we propose an improved algorithm which is more efficient than the transformation approach and Sturm's sequence method.

We have successfully used those basically mathematical methods to tackle two fundamental and difficult problems in Computer Aided Geometric Design (CAGD) and Visualization, namely approximation and representation of intersections of surfaces in three dimensions and in higher dimensions. The main contributions of our hybrid symbolic-numerical methods in this area are that: first, we solve the problem of finding a starting point for each branch of the algebraic set; secondly, we find a compromise between the two main goals of the method, namely reliability and efficiency, by using symbolic computations when needed, utilizing floating-point and parallel arithmetic whenever possible; and finally, we prevent duplicate tracing, additional singularities and reduce the degree of the defining polynomials by using pseudoirreducible decomposition.

We have implemented all of the above methods by using the kernel of the Maple language. We have also incorporated their implementations into CASA system, a computer algebra system for constructive algebraic geometry, which has been developed by the Computer Algebra group at RISC-Linz.

Zusammenfassung

Computer Aided Geometric Desgin (CAGD) und Visualisierung sind wichtige Bestandteile in der Konstruktion verschiedenster Produkte wie Automobile, Schiffe, Flugzeuge, usw. Eine anwendungsorientierte Implementierung sollte zwei Ziele erfüllen, Effizienz und Zuverlässigkeit. Numerische Algorithmen sind im allgemeinen effizient, aber nicht immer zuverlässig und können unter Umständen, wie zum Beispiel in der Nähe von Singularitäten, sogar völlig versagen. Anderseits sind Algorithmen, die auf exakter Arithmetik basieren, zuverlässig, für gewöhnlich aber langsam und speicherplatzintensiv. Effizienz und Zuverlässigkeit können wahrscheinlich nicht ohne gewisse Kompromisse gemeinsam erreicht werden.

In dieser Dissertation versuchen wir, beide Ziele ausgewogen zu berücksichtigen. Um dies zu erreichen, wurden einige hybride symbolisch-numerische Methoden entwickelt: (a) Erweitertes Newton-Verfahren. Dies ist ein neues Verfahren zur iterativen Berechnung der Lösungen eines beliebigen Gleichungssystems, wobei die Anzahl der Gleichungen größer, gleich oder kleiner als die Anzahl der Variablen sein kann. (b) Quasi-irreduzible Dekomposition. Das Ziel ist es, die aufwendige Berechnung von irreduziblen Dekompositionen, die häufig in der algebraischen Geometrie, im Computer Aided Geometric Design, usw. zur Parametrisierung, Berechnung des Geschlechts, Projektion einer algebraischen Menge auf eine Hyperebene verwendet werden, zu vermeiden. Wir schlagen ein neues Konzept, das der quasi-irreduziblen Dekomposition, vor, welches eine Verallgemeinerung der irreduziblen Dekomposition ist, aber einfacher berechnet werden kann. Es zeigt sich, daß diese Art der Dekomposition viele Anwendungen im Computer Aided Geometric Design sowie in der Visualisierung, im speziellen für eindimensionale algebraische Mengen, hat. (c) Erweiterte Dixon Resultante. Mit Hilfe der Theorie der Pseudoinversen erweitern wir die Dixon Resultante, die es erlaubt gleichzeitig mehrere Variablen eines Systems, bestehend aus $n+1$ Polynomen in n Variablen, zu eliminieren. Die Dixon Matrix kann dabei singulär oder sogar nicht-quadratisch sein kann. (d) Verbesserter Algorithmus zum Isolieren reeller Nullstellen von Polynomen mit reell-algebraischen Koeffizienten. Durch Vermeidung von Transformationen in das Einheitsintervall und Verwendung von Intervall-Arithmetik, erhalten wir einen verbesserten Algorithmus, der effizienter ist als die Transformationsmethode oder die Methode der Sturmschen Ketten.

Die oben beschriebenen Methoden wurden erfolgreich verwendet, um zwei grundlegende Probleme im Computer Aided Geometric Desgin (CAGD) und der Visualisierung zu behandeln, nämlich die Approximation und Darstellung des Durchschnittes von Flächen in drei oder mehr Dimensionen. Die wichtigsten Beiträge dieser symbolisch-numerischen Methoden dazu sind: Erstens, es wird das Problem des Auffindens eines Startpunktes für jeden Zweig einer algebraischen Menge gelöst. Zweitens, die beiden Ziele, Zuverlässigkeit und Effizienz, werden insofern berücksichtigt, als symbolische Berechnugen durchgeführt werden, wenn immer diese nötig sind und Fließkomma- und Parallel-Arithmetik verwendet wird, wenn immer dies möglich ist. Weiters werden ein mehrfaches Durchslaufen ein und desselben Zweiges und die Einführung zusätzlicher Singularitäten vermieden, wie auch der Grad der definierenden Polynome durch Verwendung quasi-irreduzibler Dekomposition reduziert wird.

Alle oben genannten Methoden wurden in Maple implementiert und in CASA, ein System für konstruktive algebraische Geometrie, entwickelt von der Computer Algebra Gruppe am RISC-Linz, eingebunden.

Contents

Preface

Computer Aided Geometric Design (CAGD) and Visualization are concerned with the approximation and representation of curves and surfaces that arise when these objects have to be processed by computer. They play an important role not only in the construction of such varied products as car bodies, ship hulls, airplane fuselages and wings, propeller blades, shoe insole, bottles, etc., but also in the description of geological, physical and even medical phenomena (see [Bar85, BFK84, Far88, Hof89, Nie93]).

A practical implementation has to balance two conflicting goals: efficiency and reliability. Roughly speaking, the algorithms must not be subject to failure and must find all the parts of the algebraic set. On the other hand, algorithms must be efficient enough so that interaction between a CAGD user and a CAGD system can be achieved. Typically, a numerical algorithm is efficient, but it is not reliable and may fail in certain cases, especially in areas near the singularities. On the other hand, algorithms based on exact arithmetic are reliable, but are normally slow and require a lot of memory space.

In spite of much effort, no present algorithm satisfies all those two main goals (see [Hof89, Pat93, Tau94a, Tau94b]). Perhaps the goals of efficiency and reliability cannot be met simultaneously without some compromises.

In this dissertation, we try to negotiate those compromises judiciously. We have developed several hybrid symbolic-numerical methods as the basically mathematical methods to achieve our goals. Among them are:

(**a**) Extended Newton's method. By using a combination of techniques from linear algebra, pseudoinverse theory and Ritt's characteristic set method, we propose a new method for iteratively determining the roots of an arbitrary system of equations, where the equations can be nonlinear algebraic or transcendental, the number of equations of the system can be greater than, less than or equal to the number of variables. In case the number of equations is equal to the number of variables, the method is similar to Newton's method.

(**b**) Pseudoirreducible decomposition. Irreducible decomposition is widely used in Algebraic Geometry, Computer Aided Geometric Design, etc. In fact, it has been used as the first and requisite step in many constructive computations such as parameterization, genus computation, projection of an algebraic set onto a hypersurface. However, irreducible decomposition is very expensive to compute from both time and memory space considerations, especially in higher dimensional spaces. In this dissertation, by using a combination of techniques from dimension theory, algebraic geometry and Ritt's characteristic set method, we propose a new concept of pseudoirreducible decomposition which is an approximation of the irreducible decomposition and easy to compute. It turns out that this new decomposition is very useful in Computer Aided Geometric Design (CAGD) and Visualization, in particular for one-dimensional algebraic sets. For example, the intersection of two parametric surfaces which can be seen as a space curve in a 4-dimensional space and the intersection of an implicit surface with an r-offset surface of another implicit surface which can be seen as a space curve in a 6-dimensional space.

(**c**) Extended Dixon's resultant. By using techniques from pseudoinverse theory, we extend Dixon's resultant for simultaneously eliminating several variables from a system of $n+1$ polynomials in n variables, where the Dixon matrix can be singular or even nonsquare.

(**d**) An improved algorithm for real root isolation of polynomials with real algebraic number coefficients. By avoiding the use of unit transformation and utilizing interval arithmetic, we propose an improved algorithm for real root isolation of polynomials with real algebraic number coefficients. The algorithm is more efficient than the transformation approach and Sturm's sequence method.

We restrict attention to algebraic curves and surfaces. This restriction is reasonable in the sense that the class is very rich and includes most of the curves and surfaces used in, for instance, engineering design. In particular, this class includes all the major parametric surfaces used in geometric modeling, including Bezier surfaces, nonuniform rational B-spline (NURBS), and so on (see [Hof89, Baj90b, BR90, Baj90a, BX94b, BX94a, BCX94]). It does not include all space curves and surfaces, however. For example, the helix is not an algebraic space curve.

We have successfully used those basically mathematical methods to tackle two fundamental and difficult problems in Computer Aided Geometric Design (CAGD) and Visualization, namely approximation and representation of intersections of surfaces in three dimensions and in higher dimensions. The main contributions of our hybrid symbolic-numerical methods in this area are that: first, we solve the problem of finding a starting point for each branch of the algebraic set; secondly, we find a compromise between the two main goals of the method, namely reliability and efficiency, by using symbolic computations when needed, utilizing floating-point and parallel arithmetic whenever possible; and finally, we prevent duplicate tracing, additional singularities and reduce the degree of the defining polynomials by using pseudoirreducible decomposition.

We have implemented all of the above methods by using the kernel of Maple language. We have also incorporated their implementations into CASA system, a computer algebra system for constructive algebraic geometry, which has been developed by the Computer Algebra group at RISC-Linz.

The structure of the dissertation is as follows: In Chapter 1, we summarize some preliminaries about the pseudoinverse of an arbitrary matrix and the continuation method, a simple and reliable numerical method for solving small zero-dimensional polynomial systems. Since there are a lot of different definitions of characteristic sets in the literature, in Section 2.1 of Chapter 2, we summarize some basic facts on Ritt's characteristic set method for the following works in this dissertation. In Section 2.2, we present some new results of the author's in an effort to enlarge the applicability of the characteristic set method to problems in Computer Aided Geometric Design and Visualization. In Section 2.3, we present an improved algorithm for real root isolation of polynomials with real algebraic number coefficients. In Chapter 3, We present a new result of the author's to extend Dixon's resultant for simultaneously eliminating several variables from a system of $n+1$ polynomials in n variables, where the Dixon matrix can be singular or even nonsquare. In Chapter 4, we present a new method of the author's for iteratively determining the roots of an arbitrary system of equations, where the number of equations of the system can be greater than, less than or equal to the number of variables. In the last two chapters, we demonstrate the ability of our hybrid symbolic-numerical methods to tackle two fundamental and difficult problems in Computer Aided Geometric Design (CAGD) and Visualization, namely approximation and representation of intersections of surfaces in three dimensions and in higher dimensions.

Lời tựa

Kính tặng Bố: Trần Quốc–Diếm

Tôi xin đặt bản Luận án Tiến Sỹ Khoa Học Kỹ Thuật nghành Khoa Học Tin Học này lên nấm mộ cha tôi – Người đã cống hiến cả cuộc đời mình cho sự nghiệp giáo dục và đào tạo thế hệ trẻ Việt nam đặc biệt là tại tỉnh Tây Ninh.

Computer Aided Geometric Design (CAGD) và Visualization là nghành khoa học ứng dụng các phương pháp Toán và Khoa Học Tin Học trong việc thiết kế các sản phẩm công nghiệp. Nó đặc biệt có hiệu quả trong công nghiệp Ô tô và công nghiệp hàng không.

Điều kiện tiên quyết cho tính khả thi của các sản phẩm phần mềm trong nghành khoa học này là: tính tin cậy và tính hiệu quả. Nói rõ hơn, các thuật toán không thể "fail" và phải tìm ra tất cả các phần của lời giải. Mặt khác, thuật toán phải có hiệu quả cao, đủ nhanh để có thể thiết lập được sự đối thoại giữa người và máy.

Thông thường, các thuật toán số có tốc độ tính toán nhanh nhưng thường không ổn định và có thể "fail" trong một số trường hợp – đặc biệt là trong các khu vực gần các điểm "singularity". Trong khi đó, các thuật toán dựa trên các kỹ thuật tính toán phi sai số (Symbolic Computation) rất ổn định nhưng thường là chậm và chiếm nhiều bộ nhớ.

Tuy đã có rất nhiều nỗ lực của các nhà nghiên cứu, nhưng cho đến nay vẫn chưa có phương pháp hay thuật toán nào đáp ứng được cả hai điều kiện tiên quyết nói trên. Có lẽ chúng ta không thể nào đạt được cả hai điều kiện đối nghịch đó (hiệu quả và tin cậy) mà không có một sự thỏa hiệp cần thiết.

Trong bản Luận án này, bằng các thuật toán lai giữa các phương pháp số và các phương pháp phi sai số hay còn gọi là các phương pháp chính xác tuyệt đối (exact methods or symbolic methods), tác giả chứng minh rằng chúng ta có thể đạt đến một sự thỏa hiệp cho cả hai điều kiện: tính hiệu quả và tính tin cậy cho các thuật toán.

Part I

Mathematical Methods

Chapter 1

Preliminaries for Linear Algebra and Numerical Methods

1.1 The Theory of Pseudoinverses

It is well known that a matrix has an inverse only if it is square and nonsingular, or in other words, its columns (or rows) are linearly independent. However, our works in Computer Aided Geometric Design (CAGD) and Visualization show that for overcoming some fundamental obstacles of the topics, we need some kinds of partial inverse of a matrix that is singular or even nonsquare. In this section, we summarize some facts about the pseudoinverse of an arbitrary matrix.

1.1.1 Introduction

Probably the most familiar application of matrices is to the solution of systems of simultaneous linear equations. Let

$$Ax = b$$

be such a system, where b is a given vector and x is an unknown vector. If A is nonsingular then there exists a unique solution for x, given by

$$x = A^{-1}b.$$

In the general case, when A may be singular or nonsquare, there may sometimes be no solutions or a multiplicity of solutions. If the system has many solutions, we may desire to have not just one solution but a characterization of all solutions. It is easy to verify that if X is any matrix satisfying $AXA = A$, then $Ax = b$ has a solution if and only if

$$AXb = b,$$

and in this case the most general solution is

$$x = Xb + (I - XA)y,$$

where y is arbitrary. We will call such a matrix X a generalize inverse of A. Unlike the case of the nonsingular matrix, which has a single unique inverse for all purposes, there are different generalized inverses for different purposes. However, in this dissertation, we are only interested in a generalized inverse whose existence is unique for every matrix (square or nonsquare), namely the Moore–Penrose generalized inverse or pseudoinverse.

The concept of generalized inverses seems to have been first mentioned in print in 1903 by Fredholm [Fre03], where a particular generalized inverse (called by him pseudoinverse) of an integral operator was given. The class of all generalized inverses was characterized in 1912 by Hurwitz [Hur12], who used the finite dimensionality of the null spaces of the Fredholm operators to give a simple algebraic construction. Generalized inverses of differential operators, already implicit in Hilbert's discussion in 1904 of generalized Green's functions [Hil12], were consequently studied by numerous authors (see [Rei68]). Pseudoinverses of matrices, whose existence was first noted by E. H. Moore (see [Moo20]), who defined a unique inverse (called by him the general reciprocal) for every finite matrix (square or nonsquare). In 1950, Bjerhammar rediscovered Moore's pseudoinverse and also noted the relationship of generalized inverses to solutions of linear systems (see [Bje51b], [Bje51a], [Bje68]). In 1955, Penrose [Pen55] sharpened and extended Bjerhammar's results on linear systems and showed that Moore's pseudoinverse, for a given matrix A, is the unique matrix A^+ satisfying the four equations (1.1)–(1.4) of the next section. The latter discovery has been so important and fruitful that this unique inverse is now commonly called the Moore–Penrose inverse or pseudoinverse.

1.1.2 Definitions and Fundamental Properties

All matrices in this section are defined over the complex number field. Clearly, analogous results can be obtained by assuming the matrices are defined over the real number field.

The following matrix equations are used to define generalized inverses of matrices where $()^*$ denotes the conjugate transpose of the matrix:

$$\begin{aligned} AXA &= A, && (1.1)\\ XAX &= X, && (1.2)\\ (AX)^* &= AX, && (1.3)\\ (XA)^* &= XA. && (1.4)\end{aligned}$$

Definition 1.1.1 *A* generalized inverse *of a matrix $A \in \mathbb{C}^{m\times n}$ is a matrix $X \in \mathbb{C}^{n\times m}$ satisfying (1.1). We denote any such matrix by $A^{(1)}$.*

Definition 1.1.2 *A* pseudoinverse *of a matrix $A \in \mathbb{C}^{m\times n}$ is a matrix $X \in \mathbb{C}^{n\times m}$, denoted in particular by A^+, satisfying (1.1), (1.2), (1.3) and (1.4).*

Given a matrix A of rank r, one can transform A to the form $\begin{pmatrix} I_r & O \\ O & O \end{pmatrix}$ by elementary row and column operations. So there exist nonsingular matrices P and Q such that

$$PAQ = \begin{pmatrix} I_r & O \\ O & O \end{pmatrix}.$$

Letting

$$X = Q\begin{pmatrix} I_r & O \\ O & W \end{pmatrix}P,$$

where W is arbitrary, it is easy to verify that $AXA = A$. In fact, every generalized inverse matrix of A can be written in this form (see [BIG74], page 17). Then we have the following lemma on the existence of generalized inverse matrices.

Lemma 1.1.1 *For any matrix A, there exists a generalized inverse $A^{(1)}$.*

For any scalar λ, we define λ^+ by

$$\lambda^+ = \begin{cases} \lambda^{-1} & (\lambda \neq 0) \\ 0 & (\lambda = 0) \end{cases}$$

For any matrix $A \in \mathbb{C}^{m\times n}$, we denote by

- $\mathcal{R}(A) = \{y \in \mathbb{C}^m : y = Ax,$ for some $x \in \mathcal{C}^n\}$, the *range* of A.
- $\mathcal{N}(A) = \{x \in \mathcal{C}^n : Ax = 0\}$, the *null space* of A.

We will observe that idempotent and Hermitian matrices are intimately related to generalized inverses. (Recall that a square matrix A is called *idempotent* if $A^2 = A$ and *Hermitian* if $A^* = A$.)

In the following lemma, we give certain properties of generalized inverses. The proof of the lemma is straightforward from the construction process of the generalized inverses and can be found in [BO71] or [BIG74].

Lemma 1.1.2 *Let $A \in \mathbb{C}^{m\times n}$, $\lambda \in \mathbb{C}$. Then*

1. rank $A^{(1)} \geq$ rank A.
2. *One choice of* $(A^*)^{(1)}$ *is* $(A^{(1)})^*$.
3. *One choice of* $(\lambda A)^{(1)}$ *is* $\lambda^+ A^{(1)}$.
4. $AA^{(1)}$ *and* $A^{(1)}A$ *are idempotent and have the same rank as* A.
5. $A^{(1)}A = I_n$ *if and only if* A *is of full column rank.*
6. $AA^{(1)} = I_m$ *if and only if* A *is of full row rank.*
7. $A^*(AA^*)^{(1)}A$ *and* $A(A^*A)^{(1)}A^*$ *are idempotent and Hermitian.*

Given a matrix $A \in \mathbb{C}^{m\times n}$. Let $X = A^{(1,2,4)}.A.A^{(1,2,3)}$, where $A^{(1,2,4)} = A^*(AA^*)^{(1)}$ and $A^{(1,2,3)} = (A^*A)^{(1)}A^*$. The identity $(BAA^* - CAA^*).(B-C)^* = (BA - CA).(BA-CA)^*$ along with the fact that $AA^* = 0$ implies $A = 0$ gives us the following result:

$$\text{if } BAA^* = CAA^* \text{ then } BA = CA. \tag{1.5}$$

Similarly, we have

$$\begin{aligned} &\text{if } BA^*A = CA^*A \quad \text{then } BA^* = CA^* && (1.6)\\ &\text{if } A^*AB = A^*AC \quad \text{then } AB = AC && (1.7)\\ &\text{if } AA^*B = AA^*C \quad \text{then } A^*B = A^*C && (1.8)\end{aligned}$$

By using (1.5), (1.6), (1.7) and (1.8), it is easy to verify that $A^{(1,2,4)}$ satisfies (1.1), (1.2) and (1.4); $A^{(1,2,3)}$ satisfies (1.1), (1.2) and (1.3); and X satisfies (1.1), (1.2), (1.3) and (1.4). Moreover, if X and Y are two matrices satisfying (1.1), (1.2), (1.3) and (1.4). Then

$$\begin{aligned} X &= X(AX)^* = XX^*A^* = X(AX)^*(AY)^* \\ &= XAY = (XA)^*(YA)^*Y = A^*Y^*Y = (YA)^*Y \\ &= Y. \end{aligned}$$

Therefore, we have the following theorem on the existence and uniqueness of the pseudoinverse of a matrix.

Theorem 1.1.1 *For any matrix A there exists a pseudoinverse matrix A^+. Moreover, the pseudoinverse is unique.*

In the following lemma, we give certain properties of the pseudoinverse of a matrix A. The proof of the lemma is straightforward and can be found in [BO71].

Lemma 1.1.3 *Let $A \in \mathbb{C}^{m \times n}$, $\lambda \in \mathbb{C}$. Then*

1. $(A^+)^+ = A$,
2. $(A^*)^+ = (A^+)^*$,
3. $(AA^*)^+ = (A^+)^* A^+$ *and* $(A^*A)^+ = A^+(A^+)^*$.
4. $\mathrm{rank}\ A^+ = \mathrm{rank}\ A = \mathrm{rank}\ A^+A$.
5. $(\lambda A)^+ = \lambda^+ A^+$.
6. $A^+ = (A^*A)^+ A^* = A^*(AA^*)^+$.
7. AA^+, A^+A, $I - AA^+$ *and* $I - A^+A$ *are Hermitian and idempotent.*
8. $A^+A = I_n$ *if and only if A is of full column rank.*
9. $AA^+ = I_m$ *if and only if A is of full row rank.*

Note that, in general we may not have $(AB)^+ = B^+A^+$.

1.1.3 Computing Techniques

In this subsection, we summarize some known techniques for obtaining the pseudoinverse of a matrix. We also give an algorithm for iteratively obtaining the pseudoinverse.

Direct Methods

Given a nonnull matrix A of rank r. Let F be any matrix whose columns are a basis for the column space of A. Then F is a matrix of full column rank. Since every column of A is uniquely representable as a linear combination of the columns of F, there exists a unique matrix G such that $A = FG$ and G is a matrix of full row rank. Alternatively, we can first start with a matrix G' of full row rank whose rows are a basis for the row space of A. Then there uniquely exists a matrix F' such that $A = F'G'$ where F' is a matrix of full column rank. Therefore, we have the following lemma.

Lemma 1.1.4 *Let A be a matrix of rank r, $r > 0$ (i.e. $A \in \mathbb{C}_r^{m \times n}$). Then there exist matrices F of full column rank (i.e. $F \in \mathbb{C}_r^{m \times r}$) and G of full row rank (i.e. $G \in \mathbb{C}_r^{r \times n}$), such that*

$$A = FG$$

We will call such a factorization a *full-rank factorization* of A. It turns out that full-rank factorization is a powerful tool in the study of pseudoinverses. In fact, a full-rank factorization of a matrix A leads to an explicit formula for its pseudoinverse A^+.

Lemma 1.1.5 *Let $A = FG$ be a full-rank factorization of a matrix $A \in \mathbb{C}_r^{m \times n}$, $r > 0$, where $F \in \mathbb{C}_r^{m \times r}$ and $G \in \mathbb{C}_r^{r \times n}$. Then*

$$A^+ = G^*(GG^*)^{-1}(F^*F)^{-1}F^* = G^+F^+ \tag{1.9}$$

The proof of the lemma is simple and can easily be verified by just checking that the right hand side of Equation (1.9) satisfies Equation (1.1), (1.2), (1.3) and (1.4). It can also be found in [BIG74].

Iterative Methods

In computing the pseudoinverse of a matrix $A \in \mathbb{C}^{m\times n}$, the size or difficulty of the problem can be reduced if A is considered as a block matrix

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{12} & A_{22} \end{bmatrix}$$

and its pseudoinverse is expressed in terms of the submatrices A_{ij}.

In [Gre60], Greville presented a concise recursive algorithm for computing the pseudoinverse of a matrix $A \in \mathbb{C}^{m\times n}$, $n \geq 2$. The method is based on partitioning A as

$$A = \begin{bmatrix} A_{n-1} & a_n \end{bmatrix}$$

where a_n is the nth column and A_{n-1} is the submatrix consisting the first $n-1$ columns of A. The pseudoinverse of A, namely A^+, is then expressed in terms of a_n and A^+_{n-1}, which is iteratively computed in the same way, using the partition

$$A_{n-1} = \begin{bmatrix} A_{n-2} & a_{n-1} \end{bmatrix}, \quad \text{etc.}$$

The following theorem has been proved in [Gre60].

Theorem 1.1.2 (Greville) *Let $A \in \mathbb{C}^{m\times n}$. The pseudoinverse of $A_k, k = 2\ldots n$, is*

$$\begin{bmatrix} A_{k-1} & a_k \end{bmatrix}^+ = \begin{bmatrix} A^+_{k-1} - d_k b_k \\ b_k \end{bmatrix}$$

where

$$\begin{aligned} d_k &= A^+_{k-1} a_k \\ c_k &= a_k - A_{k-1} d_k \\ b_k &= \begin{cases} c_k^+ & \text{if } c_k \neq 0 \\ (1 + d_k^* d_k)^{-1} d_k^* A^+_{k-1} & \text{if } c_k = 0 \end{cases} \end{aligned}$$

This leads to an algorithm for iteratively obtaining the pseudoinverse of a matrix as follows.

Algorithm 1.1.1 (Iterative-Method)

Input: A matrix $A \in \mathbb{C}^{m\times k}$

Output: The pseudoinverse A^+ of A.

1. **if** $k = 1$ *then*
   if $a_1 = 0$ *then* **Return**(0)
   else Return$((a^* a)^{-1} a^*)$

2. $A^+_{k-1} =$ ***Iterative-Method***(A_{k-1})

3. $d_k = A^+_{k-1} a_k$

4. $c_k = a_k - A_{k-1} d_k$

5. **if** $c_k = 0$ *then* $b_k = (1 + d_k^* d_k)^{-1} d_k^* A^+_{k-1}$
   else $b_k = c_k^+$

6. $A^+ = \begin{bmatrix} A^+_{k-1} - d_k b_k \\ b_k \end{bmatrix}$

7. **Return**(A^+)

1.2 Continuation Method

The continuation method is a simple, reliable numerical method for solving small zero-dimensional polynomial systems. The method differs from more classical techniques in that it is *global* and *exhaustive*. It does not require a choice of initial solution estimates, and it finds all solutions. This is in contrast to local methods, such as Newton's method.

Given a system of n equations in n variables, $f(x) = 0$, where the algebraic set of the system is zero-dimensional. An initial system $g(x) = 0$ is created with known roots and the same total degree as $f(x)$, for example:

$$g_i(x) = p_i^{d_i} x_i^{d_i} - q_i^{d_i}$$

for $i = 1, \ldots, n$, $d_i = \deg(f_i)$ and p_i, q_i are complex constants to be chosen randomly. Then $g(x) = 0$ has $d = \prod_{i=1}^{n} d_i$ solution, $x^{(0)}$, given by

$$x^{(0)} = \left(u(d_1, k_1)\frac{q_1}{p_1}, u(d_2, k_2)\frac{q_2}{p_2}, \ldots, u(d_n, k_n)\frac{q_n}{p_n} \right)$$

for $1 \leq k_1 \leq d_1$, $1 \leq k_2 \leq d_2, \ldots$, $1 \leq k_n \leq d_n$, where u gives the roots of unity defined by

$$u(d, k) = \cos(\frac{2\pi k}{d}) + i \sin(\frac{2\pi k}{d}).$$

Then the continuation, parameterized by t, between g and f is:

$$h(x, t) = (1 - t)g(x) + tf(x).$$

This continuation generates d paths, beginning when $t = 0$ at each of the solutions of $g(x) = 0$. The roots of $f(x) = 0$ are found by tracing those paths.

The correctness of the continuation method is guaranteed by the following theorem (see [Mor87], page 60):

Theorem 1.2.1 *Given f, there are sets of measure zero, A_p and A_q, in $\mathbb{C}^n$ so that if $p \notin A_p$ and $q \notin A_q$, then*

1. *The solution set $\{(x, t) \in \mathbb{C}^n \times [0, 1) : h(x, t) = 0\}$ is a collection of d nonoverlapping (smooth) paths.*
2. *The paths move from $t = 0$ to $t = 1$ without backtracking in t.*
3. *Each geometrically isolated solution of $f(x) = 0$ of multiplicity m has exactly m continuation paths converging to it.*
4. *A continuation path can diverge to infinity only as $t \to 1$.*
5. *If $f(x) = 0$ has no solution at infinity, all the paths remain bounded. Otherwise, at least one path will diverge to infinity as $t \to 1$. Each geometrically isolated solution at infinity of $f(x) = 0$ of multiplicity m will generate exactly m diverging continuation paths.*

Details of this method can be found in [Mor87, Dre78].

Chapter 2

Preliminaries for Symbolic Methods

2.1 Characteristic Set Method

The characteristic set method was completely developed by J. F. Ritt in the early 1930s (see [Rit32], [Rit50]) and has been successfully used by W. T. Wu and his followers for mechanical theorem proving in geometry (see [Wu84, Wu94], [Cho88] and [Wan94]). Since there are a lot of different definitions of characteristic sets in the literature, in this section we summarize some basic facts on Ritt's characteristic set method for the following works in this dissertation. Additionally, in the next section, we present some new results of the author's in an effort to enlarge the applicability of the characteristic set method to problems in Computer Aided Geometric Design and Visualization. More details can be found in [Tra96c].

2.1.1 Basic Definitions

Let K be a computable field and $\tilde{x} = (x_1, x_2, \ldots, x_k)$ be indeterminates with a total order $x_1 \prec x_2 \prec \ldots \prec x_k$. All polynomials mentioned in this section are in $K[\tilde{x}] = K[x_1, x_2, \ldots, x_k]$, the ring of all polynomials with coefficients in K.

Definition 2.1.1 *Let f_1, f_2, ..., f_h be polynomials in $K[\tilde{x}]$. We call $\mathcal{V}(f_1, f_2, \ldots, f_h)$ the affine* algebraic set *or the* zero set *defined by f_1, f_2, ..., f_h where*

$$\mathcal{V}(f_1, f_2, \ldots, f_h) = \{(\zeta_1, \zeta_2, \ldots, \zeta_k) \in K^k : f_i(\zeta_1, \zeta_2, \ldots, \zeta_k) = 0, \text{ for all } 1 \le i \le h\}.$$

Definition 2.1.2 *Let $f \in K[\tilde{x}]$. The* class *of f, denoted by* $\mathrm{class}(f)$, *is the largest i such that x_i occurs in f. If $f \in K$, i.e. there is no variable, then* $\mathrm{class}(f) = 0$. *If* $\mathrm{class}(f) = c > 0$, *we call x_c, denoted by* $\mathrm{lv}(f)$, *the* leading variable *of f.*

Definition 2.1.3 *Consider f as a polynomial in x_c, the leading variable of f. We can write f as:*

$$a_n x_c^n + \ldots + a_1 x_c + a_0$$

where $a_n, a_{n-1}, \ldots a_0$ are in $K[x_1, x_2, \ldots x_{c-1}]$, $n > 0$ and $a_n \neq 0$. We call a_n the initial *or* leading coefficient *of f and n the* leading degree *of f, denoted as* $\mathrm{lc}(f)$ *and* $\mathrm{ld}(f)$, *respectively.*

Example 2.1.1 *Let f be*

$$f = 1 + ((1 + 2x_1)x_2 + x_1 x_2^3)x_3 + (1 + x_1^3 x_2 + (1 + 2x_1^2)x_2^2)x_3^2$$

Then $\mathrm{class}(f) = 3$, $\mathrm{lv}(f) = x_3$, $\mathrm{lc}(f) = 1 + x_1^3 x_2 + (1 + 2x_1^2)x_2^2$ *and* $\mathrm{ld}(f) = 2$.

Definition 2.1.4 *For any variable $x_i \in \{x_1, x_2, \ldots x_k\}$, we can consider the non-zero polynomial f as a polynomial in x_i, and write f as*

$$b_m x_i^m + \ldots + b_1 x_i + b_0,$$

where $b_m, b_{m-1}, \ldots b_0$ are in $K[x_1, \ldots x_{i-1}, x_{i+1}, \ldots x_k]$, $m > 0$ and $b_m \neq 0$. We call m the degree of f with respect to x_i, denoted by $\deg(f, x_i)$.

For the use in the following parts of our dissertation, we next present an algorithm for pseudodivision – a generalized arithmetic for division of polynomials over any algebraic system that is a commutative ring with identity. Pseudodivision also is a basic step for Ritt's characteristic set method.

Theorem 2.1.1 *Let g and f be multivariate polynomials in $K[\tilde{x}]$ and x_i be one of the variables $x_1, x_2, \ldots x_k$. Suppose that $\deg(g, x_i) \geq \deg(f, x_i) > 0$. We regard g and f as polynomials in x_i:*

$$g = a_n x_i^n + \ldots + a_1 x_i + a_0, \quad f = b_l x_i^l + \ldots + b_1 x_i + b_0.$$

Then there are uniquely defined polynomials q and r in x_i and $m \in \{0, \ldots, n-l+1\}$ such that

$$b_l^m . g = q . f + r, \tag{2.1}$$

where m is minimal and $\deg(r, x_i) < \deg(f, x_i)$ or $r = 0$.

Proof. (See [Win96]). □

Definition 2.1.5 *Let K, g, f, x_i, n, l be as in Theorem 2.1.1. Then the uniquely defined polynomials $q(x_i)$ and $r(x_i)$ satisfying (2.1) are called the* pseudoquotient *and* pseudoremainder *of $g(x_i)$ with respect to $f(x_i)$, denoted by $\mathrm{pquot}(g, f, x_i)$ and $\mathrm{prem}(g, f, x_i)$ respectively.*

Algorithm 2.1.1 (pseudodivision)

Inputs: $g, f \in K[\tilde{x}]$ where $\deg(g, x_i) \geq \deg(f, x_i) > 0$.

Outputs: $q(x_i) = \mathrm{pquot}(g, f, x_i)$ and $r(x_i) = \mathrm{prem}(g, f, x_i)$.

1. $r = g$; $q = 0$.

2. **while** $m = deg(r, x_i) \geq l$ **do**
   $r = b_l . r - c_m . x_i^{m-l} . f$
   $q = b_l . q + c_m . x_i^{m-l}$

where c_m is the leading coefficient of r in the variable x_i.

It is easy to see that m strictly decreases after each iteration, hence the process terminates.

Example 2.1.2 *Let*

$$g = 1 + ((1 + 2x_1)x_2 + x_1 x_2^3)x_3 + (1 + x_1^3 x_2 + (1 + 2x_1^2)x_2^2)x_3^5$$

and let

$$f = 5 + x_1 x_2 + (3 + x_1^2 + x_2^3)x_3^2$$

We will get

$$
\begin{aligned}
r \;=\;& 9 + 6x_1^2 + x_1^4 + 6x_2^3 + 2x_1^2x_2^3 + x_2^6 + 25x_3 + 9x_2x_3 + 28x_1x_2x_3 \\
& + 6x_1^2x_2x_3 + 37x_1^3x_2x_3 + x_1^4x_2x_3 + 2x_1^5x_2x_3 + 25x_2^2x_3 + 51x_1^2x_2^2x_3 \\
& + 10x_1^4x_2^2x_3 + 19x_1x_2^3x_3 + 26x_1^3x_2^3x_3 + 2x_1^5x_2^3x_3 + 6x_2^4x_3 + 12x_1x_2^4x_3 \\
& + 3x_1^2x_2^4x_3 + 4x_1^3x_2^4x_3 + 2x_1^4x_2^4x_3 + 6x_1x_2^6x_3 + 2x_1^3x_2^6x_3 + x_2^7x_3 \\
& + 2x_1x_2^7x_3 + x_2^9x_3
\end{aligned}
$$

$$
\begin{aligned}
q \;=\;& -5x_3 - x_1x_2x_3 - 5x_1^3x_2x_3 - 5x_2^2x_3 - 10x_1^2x_2^2x_3 - x_1^4x_2^2x_3 - x_1x_2^3x_3 \\
& - 2x_1^3x_2^3x_3 + 3x_3^3 + x_1^2x_3^3 + 3x_1^3x_2x_3^3 + x_1^5x_2x_3^3 + 3x_2^2x_3^3 + 7x_1^2x_2^2x_3^3 \\
& + 2x_1^4x_2^2x_3^3 + x_2^3x_3^3 + x_1^3x_2^4x_3^3 + x_2^5x_3^3 + 2x_1^2x_2^5x_3^3
\end{aligned}
$$

$$b_l = (3 + x_1^2 + x_2^3), s = 2 \;\; \textit{and} \;\; b_l^2.g = q.f + r$$

We call a polynomial g reduced with respect to another polynomial f if $\deg(g, \mathrm{lv}(f)) < \deg(f, \mathrm{lv}(f))$. Note that $\mathrm{prem}(g, f, \mathrm{lv}(f))$ is always reduced with respect to f for any polynomial pair f and g.

2.1.2 Chains and Characteristic Sets

Definition 2.1.6 (Triangular Forms) *Let $f_1, f_2, \ldots f_h$ be a sequence of polynomials in $K[\tilde{x}]$. We call it a* triangular form *or a quasiascending chain if*

- *Either $h = 1$ and $f_1 \neq 0$,*
- *Or $h > 1$ and $0 < class(f_1) < class(f_2) < \ldots < class(f_h)$.*

In case the triangular form has the number of its polynomials equal to the number of its indeterminates, we call it a strong triangular form . .

Let f_1, f_2, ..., f_h be a triangular form with $0 < class(f_1)$. We define $\mathrm{prem}(g; f_1, f_2, \ldots, f_h)$ inductively to be $\mathrm{prem}(\mathrm{prem}(g, f_h); f_1, f_2, \ldots, f_{h-1})$. Let it be R. We then have the following important remainder formula:

$$I_1^{s_1} \ldots I_h^{s_h}.g = Q_1.f_1 + \ldots + Q_h.f_h + R \tag{2.2}$$

where I_i is the initial of f_i, $i = 1 \ldots h$; $s_1, \ldots s_h$ are nonnegative integers and $Q_1, \ldots Q_h$ are polynomials. Furthermore, $\deg(R, \mathrm{lv}(f_i)) < \deg(f_i, \mathrm{lv}(f_i))$, for $i = 1 \ldots h$.

Definition 2.1.7 (Chains) *Let $\mathcal{C} = f_1, f_2, \ldots f_h$ be a triangular form of polynomials in $K[\tilde{x}]$. $\mathcal{C}$ is called an ascending chain if f_j is reduced with respect to f_i, for $i < j$.*

Definition 2.1.8 *Let f and g be two polynomials in $K[\tilde{x}]$. We define $f \prec g$ if* $\mathrm{class}(f) < \mathrm{class}(g)$ *or* $\mathrm{class}(f) = \mathrm{class}(g)$ *and* $\mathrm{ld}(f) < \mathrm{ld}(g)$. *If neither $f \prec g$ nor $g \prec f$, then we say f and g are of the same rank, denoted by $f \sim g$.*

Obviously, the partial order $\prec$ in $K[\tilde{x}]$ defined as above is well order and therefore every nonempty set $S \subset K[\tilde{x}]$ has a minimal element.

Definition 2.1.9 *Let $\mathcal{C}_1 = f_1, \ldots, f_h$ and $\mathcal{C}_2 = g_1, \ldots, g_m$ be two ascending chains. We define $\mathcal{C}_1 \prec \mathcal{C}_2$ if*

- *Either there is an index s such that $s \leq \min(h, m)$, $f_i \sim g_i$ for $i < s$ and $f_s \prec g_s$,*
- *Or $h > m$ and $f_i \sim g_i$ for $i \leq m$.*

If neither $\mathcal{C}_1 \prec \mathcal{C}_2$ *nor* $\mathcal{C}_2 \prec \mathcal{C}_1$*, then we say* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are of the same rank, denoted by* $\mathcal{C}_1 \sim \mathcal{C}_2$.

Lemma 2.1.1 *The partial order* $\prec$ *on the set of all ascending chains is a well order, i.e. there are no infinite, strictly decreasing sequences of ascending chains.*

PROOF. (See [Rit50]). □

We next define characteristic sets and present some algorithms to construct them (more precisely, we can compute an extended characteristic set of an ideal and/or an irreducible characteristic set for the corresponding prime ideal of each irreducible component of the algebraic set defined by the given ideal). More details can be found in [Rit50], [Wu84], [Wu94], [Cho88] and [CG90].

Definition 2.1.10 (Characteristic Sets) *Let* $\mathrm{PS} = \{f_1, f_2, \ldots, f_h\} \subset K[\tilde{x}]$ *be a finite set of polynomials,* $\mathcal{I}$ *be the ideal generated by* PS *and* $\mathcal{C} \subset \mathcal{I}$ *be an ascending chain.*

- $\{\mathcal{C}\}$ *is called a characteristic set of the ideal* $\mathcal{I}$ *if* $\mathrm{prem}(f, \mathcal{C}) = 0, \ \forall f \in \mathcal{I}$.
- $\{\mathcal{C}\}$ *is called an extended characteristic set of the ideal* $\mathcal{I}$ *if* $\mathrm{prem}(f_i, \mathcal{C}) = 0$, $i = 1 \ldots h$.

Note that a characteristic set or an extended characteristic set is not unique. In fact, $\{\mathcal{C}\}$ is a characteristic set if and only if $\mathcal{C}$ is a minimal ascending chain with respect to the partial order $\prec$ on the set of all ascending chains of the ideal $\mathcal{I}$. Meanwhile, an extended characteristic set may not be a minimal ascending chain of the ideal.

Theorem 2.1.2 (Ritt) *Given a finite set* $\mathrm{PS} = \{f_1, f_2, \ldots, f_h\} \subset K[\tilde{x}]$ *of polynomials, Algorithm 2.1.2 constructs, in a finite number of steps, an extended characteristic set of the ideal generated by* PS *such that*

$$\mathcal{V}(\mathrm{CS}) \setminus \mathcal{V}(J) \subseteq \mathcal{V}(\mathrm{PS}) \subseteq \mathcal{V}(\mathrm{CS}) \tag{2.3}$$

$$\mathcal{V}(\mathrm{PS}) = (\mathcal{V}(\mathrm{CS}) \setminus \mathcal{V}(J)) \cup \bigcup_{i=1}^{h} \mathcal{V}(\mathrm{PS}_i) \tag{2.4}$$

where $I_i = \mathrm{lc}(f_i)$, $J = I_1 \cdots I_h$, $\mathrm{PS}_i = \mathrm{PS} \cup \{I_i\}$, $i = 1 \ldots h$.

In order not to confuse the readers, we suggest not to use the notation $Zero(\mathrm{CS}/J)$ for $\mathcal{V}(\mathrm{CS}) \setminus \mathcal{V}(J)$. This notation is used in algebra but with a quite different meaning, namely the zero set of an ideal quotient.

Algorithm 2.1.2 (E-Char-Set)

Input: A system PS *of multivariate polynomials.*

Output: An extended characteristic set CS *of* PS.

1. QS := PS; RS := PS
2. **while** $RS \neq \emptyset$ **do**
   CS := **Basic-Chain**(QS)
   if *a nonzero number is in* CS *then* **Return(1)**
   else
   $RS := \{r : r = \mathrm{prem}(q, \mathrm{CS}),\ r \neq 0,\ q \in \mathrm{QS} \setminus \mathrm{CS}\}$
   QS := QS $\cup$ RS

3. **Return**(CS)

Algorithm 2.1.3 (Basic-Chain)

Input: A system S of multivariate polynomials.

Output: A basic chain of S.

1. **if** *a nonzero number is in S then* **Return(1)**
   $BS := \emptyset$; $T := S$

2. **while** $T \neq \emptyset$ **do**
   $p :=$ *a smallest polynomial in T (with respect to $\prec$)*
   $BS := BS \cup \{p\}$
   $T := \{q : q \in T \setminus \{p\},\ q$ *is reduced with respect to* $p\}$

3. **Return**(BS)

If we apply Algorithm 2.1.2 recursively to the system $\mathrm{PS}_i \cup \mathrm{CS}$ where $\mathcal{V}(\mathrm{PS}_i) = \mathcal{V}(\mathrm{PS}_i \cup \mathrm{CS})$ and the rank of the extended characteristic set of PS_i is lower than that of CS, we would obtain a sequence of extended characteristic sets CS_i where $\mathcal{V}(\mathrm{PS}) = \bigcup_i \left(\mathcal{V}(\mathrm{CS}^{(i)}) \setminus \mathcal{V}(J^{(i)})\right)$ and $J^{(i)}$ is the product of the leading coefficients of polynomials in $\mathrm{CS}^{(i)}$.

2.1.3 Irreducible Characteristic Sets

Given an ascending chain $\mathcal{C}$, for convenience, let us change the variables such that $\mathcal{C}$ can be written in the form

$$\begin{array}{l} f_1(u_1, \ldots, u_{k-h}, x_1), \\ f_2(u_1, \ldots, u_{k-h}, x_1, x_2), \\ \ldots \\ f_h(u_1, \ldots, u_{k-h}, x_1, \ldots, x_h). \end{array} \tag{2.5}$$

We inductively define irreducible ascending chains together with their extended zeros as follows.

Definition 2.1.11 (Irreducible Chains) *Let $f_1, f_2, \ldots f_h$ be an ascending chain as above. We call it an* irreducible ascending chain *together with its* extended zero *ζ if*

- *Either $h = 1$ and f_1 as a polynomial in $K(u_1, \ldots, u_{k-h})[x_1]$ is irreducible. In this case, $\zeta = (u_1, \ldots, u_{k-h}, \zeta_1)$, where ζ_1 is any zero of $f_1 = 0$ in the algebraic closure of $K(u_1, \ldots, u_{k-h})$.*

- *Or $h > 1$; $f_1, \ldots, f_{h-1}$ is irreducible with an extended zero $(u_1, \ldots, u_{k-h}, \zeta_1, \ldots, \zeta_{h-1})$ and $\bar{f}_h = f_h(\zeta_1, \ldots, \zeta_{h-1}, x_h)$ as a polynomial in $K(u_1, \ldots, u_{k-h}, \zeta_1, \ldots, \zeta_{h-1})[x_h]$ is irreducible. In this case, $\zeta = (u_1, \ldots, u_{k-h}, \zeta_1, \ldots, \zeta_h)$, where ζ_h is any zero of $\bar{f}_h = 0$ in the algebraic closure of $K(u_1, \ldots, u_{k-h}, \zeta_1, \ldots, \zeta_{h-1})$.*

Lemma 2.1.2 *Given an irreducible ascending chain $\mathcal{C} = f_1, f_2, \ldots f_h$ with an extended zero $\zeta = (u_1, \ldots, u_{k-h}, \zeta_1, \ldots, \zeta_h)$ and a polynomial f in $K[u_1, \ldots, u_{k-h}, x_1, \ldots, x_{h-1}, x_h]$. Then $\mathrm{prem}(f, \mathcal{C}) = 0$ if and only if $f(\zeta) = 0$.*

PROOF. It is easy to check by using induction on the length of the ascending chain that if $f \in K[u_1, \ldots, u_{k-h}, x_1, \ldots, x_{h-1}, x_h]$ is reduced with respect to $\mathcal{C}$ and $f(\zeta) = 0$ then f is identical to zero. Let the pseudoremainder of f with respect to $\mathcal{C}$ be R then we have

$$I_1^{r_1} \cdots I_h^{r_h} \cdot f = \sum_{i=1}^{h} q_i \cdot f_i + R$$

Suppose that $R = 0$. Since ζ is an extended zero of all f_i's while it is not an extended zero of any I_i, ζ is an extended zero of f. Conversely, if ζ is an extended zero of f then it is also an extended zero of R, and hence R has to be identical to zero. $\square$

It should be stressed that an irreducible ascending chain $\mathcal{C}$ may not define a prime ideal. In other words, $\mathcal{V}(\mathcal{C})$ may be reducible. The problem comes from the fact that if there exists a transcendental variable in $\mathcal{C}$ then it may damage the primeness property of the ideal; for example, $\mathcal{C} = \{x_1(x_2^2 + 1)\}$ is an irreducible ascending chain, but obviously it does not define a prime ideal.

Theorem 2.1.3 *Let* $\mathcal{C}$, ζ *as in Lemma 2.1.2 and* $\mathrm{PD} = \{f : \mathrm{prem}(f, \mathcal{C}) = 0\}$. *Then* PD *is a prime ideal with the generic point* ζ. *Moreover* $\mathcal{C}$ *is a characteristic set of* PD.

PROOF. It is easy to prove by using Lemma 2.1.2 that PD is a prime ideal with ζ being its generic point. The fact that $\mathcal{C}$ is a characteristic set of PD comes directly from the definition of characteristic sets. $\square$

Lemma 2.1.3 *Let a set* PS *of polynomials have an extended characteristic set* CS *with* $\mathrm{class}(f_i) > 0$, $I_i = lc(f_i)$ *for* $i = 1 \ldots h$. *Suppose that* CS *is reducible. That means, there is an index* h_0 *for which the* h_0*-th polynomial* f_{h_0} *of* CS *is reducible and has an irreducible factorization into polynomials* $g_1, g_2, \ldots, g_r$. *Then we have a zero decomposition as follows.*

$$\mathcal{V}(\mathrm{PS}) = \bigcup_{i=1}^{h_0-1} \mathcal{V}(\mathrm{PS}_i) \cup \bigcup_{j=1}^{r} \mathcal{V}(\mathrm{QS}_j) \tag{2.6}$$

where $\mathrm{PS}_i = \mathrm{PS} \cup \{I_i\}$ *and* $\mathrm{QS}_j = \mathrm{PS} \cup \{g_j\}$.

PROOF. The right-hand-side of Equation (2.6) is obviously a subset of the left-hand-side. Conversely, since $f_1, f_2, \ldots f_{h_0-1}$ is an irreducible ascending chain with an extended zero, say $\zeta = (u_1, \ldots, u_{k-h}, \zeta_1, \ldots, \zeta_{h_0-1})$, we have $\bar{f}_{h_0} = g'_1 \cdot g'_2 \cdots g'_r$, where $\bar{f}_{h_0}$ is obtained from f_{h_0} by substituting $\zeta_1, \ldots, \zeta_{h_0-1}$ for $x_1, \ldots, x_{h_0-1}$ and $g'_1, g'_2, \ldots g'_r$ are in $K(u_1, \ldots, u_{k-h}, \zeta_1, \ldots, \zeta_{h_0-1})[x_{h_0}]$. Multiplying both sides by the common denominator and a suitable polynomial in $K[u_1, \ldots, u_{k-h}, \zeta_1, \ldots, \zeta_{h_0-1}]$, we get $\bar{d} \cdot \bar{f}_{h_0} = \bar{g}_1 \cdot \bar{g}_2 \cdots \bar{g}_r$ where $\bar{d}$, $\bar{g}_j$ are obtained from some d, g_j by substituting $\zeta_1, \ldots, \zeta_{h_0-1}$ for $x_1, \ldots, x_{h_0-1}$ with $d \in K[u_1, \ldots, u_{k-h}, x_1, \ldots, x_{h_0-1}]$ and $g_j \in K[u_1, \ldots, u_{k-h}, x_1, \ldots, x_{h_0-1}][x_{h_0}]$. We may assume that d is already reduced with respect to $\mathrm{CS}^{(h_0-1)}$ and g_j, $j = 1 \ldots r$, is already reduced with respect to $\mathrm{CS}^{(h_0)}$. Write $g_1 \cdot g_2 \cdots g_r - d \cdot f_{h_0}$ as a polynomial in x_{h_0}, say

$$g_1 \cdot g_2 \cdots g_r - d \cdot f_{h_0} = \sum_j a_j x_{h_0}^j, \tag{2.7}$$

where $a_j \in K[u_1, \ldots, u_{k-h}, x_1, \ldots, x_{h_0-1}]$ and $a_j(u_1, \ldots, u_{k-h}, \zeta_1, \ldots, \zeta_{h_0-1}) = 0$; therefore, all a_j have pseudoremainder zero with respect to the irreducible ascending chain $\mathrm{CS}^{(h_0-1)}$. That means, there are nonnegative integers $r_{(j,1)}, \ldots, r_{(j,h_0-1)}$ and polynomials $q_{(j,i)} \in K[u_1, \ldots, u_{k-h}, x_1, \ldots, x_{h_0-1}]$ satisfying $I_1^{r(j,1)} \cdots I_{h_0-1}^{r(j,h_0-1)} \cdot a_j = \sum_{i=1}^{h_0-1} q_{(j,i)} f_i$. Multiplying both sides of Equation (2.7) by $I_1^{\hat{r}_1} \cdots I_{h_0-1}^{\hat{r}_{h_0-1}}$, where $\hat{r}_i = max_j r_{(j,i)}$, we get $I_1^{\hat{r}_1} \cdots I_{h_0-1}^{\hat{r}_{h_0-1}} g_1 \cdots g_r = \sum_{i=1}^{h_0} \hat{q}_{(j,i)} f_i$. Since any zero of PS is a zero of the f_i's, it is also a zero of some I_i's or g_j's; and hence a zero of some PS_i's or QS_j's. □

The following breadth-first algorithm computes a sequence of irreducible extended characteristic sets of a system of polynomials.

Theorem 2.1.4 (Ritt) *Given a finite set* PS *of polynomials, Algorithm 2.1.4 constructs, in a finite number of steps, a sequence* $(\mathrm{CS}^{(i)})$ *of irreducible extended characteristic sets of the ideal generated by* PS *such that*

$$\mathcal{V}(\mathrm{PS}) = \bigcup_{i=1}^{r} \left(\mathcal{V}(\mathrm{CS}^{(i)}) \setminus \mathcal{V}(J^{(i)}) \right) \tag{2.8}$$

where $J^{(i)}$ *is the product of the leading coefficients of polynomials in* $\mathrm{CS}^{(i)}$.

Proof. (See [Rit50]). □

Algorithm 2.1.4 (Char-Set)

Input: A system PS *of multivariate polynomials.*

Output: A sequence $(\mathrm{CS}^{(i)})$ *of irreducible extended characteristic sets of* PS.

1. $\mathcal{Q} := \{\mathrm{PS}\}$; $\mathcal{P} := \emptyset$; $\mathcal{T} := \emptyset$; $\mathcal{S} := \emptyset$;
2. **while** $\mathcal{Q} \neq \emptyset$ **do**
   - (a) $\mathcal{P} := \mathcal{Q}$;
   - (b) **while** $\mathcal{P} \neq \emptyset$ **do**
     - i. QS := *first-element*($\mathcal{P}$);
     - ii. $\mathcal{P}$:= *rest*($\mathcal{P}$);
     - iii. CS := **E-Char-Set**(QS);
     - iv. **if** CS = (1) *then* **next**;
     - v. **else**
       if CS *is irreducible then*
       $\mathcal{S} := \mathcal{S} \cup \{\mathrm{CS}\}$;
       $\mathcal{T} := \mathcal{T} \cup \{\mathrm{QS} \cup \mathrm{CS} \cup \{I\} : I = \mathrm{lc}(f), f \in \mathrm{CS}, \mathrm{class}(I) > 0\}$;
       else
       Factorize certain f_{h_0} *in* CS *into* $g_1 \cdots g_r$;
       $\mathcal{T} := \mathcal{T} \cup \{\mathrm{QS} \cup \mathrm{CS} \cup \{I\} : I = \mathrm{lc}(f), f \in \mathrm{CS}, \mathrm{class}(f) < \mathrm{class}(f_{h_0})\} \cup \{\mathrm{QS} \cup \mathrm{CS} \cup \{g_j\} : 1 \leq j \leq r\}$
   - (c) $\mathcal{Q} := \mathcal{T}$; $\mathcal{T} := \emptyset$;
3. **Return**($\mathcal{S}$).

2.2 Pseudoirreducible Decomposition

Irreducible decomposition is widely used in Algebraic Geometry, Computer Aided Geometric Design, etc. In fact, it has been used as the first and requisite step in many constructive computations such as parameterization, genus computation, projection of an algebraic set onto a hypersurface. It is a well-known result that we can algorithmically decompose an algebraic set (in K^n where K is a field) into a union of finitely many irreducible and noncontractive components. The decomposition is unique up to the order in which the components are written. Algorithms for finding the decomposition can be found in [Her26], [Rit32, Rit50], [MRR88], [Sei74, Sei84], [GTZ88] and [EHV90]. Among them, the most efficient method for finding the decomposition is the method of Ritt-Wu's based on characteristic set computations. By Ritt-Wu's method, to decompose an algebraic set, we first compute a sequence of irreducible characteristic sets which have the union of their corresponding irreducible algebraic sets equal to the original one. Then to find the generating set of the corresponding irreducible algebraic set for each irreducible characteristic set, we need to compute a Gröbner basis of a system of equations which includes the characteristic set itself and an extra equation with an extra variable. However, for algebraic sets in higher dimensional spaces, such the Gröbner bases are very expensive to compute from both time and memory space considerations. Meanwhile in practise, for example in Computer Aided Geometric Design (CAGD) or Visualization, we need only an approximation of the algebraic set, in particular for one-dimensional algebraic sets. In fact, approximation and representation of one-dimensional algebraic sets in higher dimensional space is a fundamental and difficult problem in Computer Aided Geometric Design (CAGD) and solid modeling. For example, the intersection of two parametric surfaces can be seen as a space curve in a 4-dimensional space and the intersection of an implicit surface with an r-offset surface of another implicit surface can be seen as a space curve in a 6-dimensional space.

In this section, we present some new results of the author's concerning the measure of the corresponding irreducible components of an algebraic set in an effort to enlarge the applicability of the characteristic set method to problems in Computer Aided Geometric Design and Visualization. But first of all, we need some basic theorems from Ritt-Wu's method to decompose an algebraic set. Proofs of the following theorems are straightforward and can be found in [Cho88] and [Wan94].

Lemma 2.2.1 *Let* $\mathrm{CS}^{(i)}$ *be as in Theorem 2.1.4 and* $\mathrm{PD}^{(i)} = \{f : \mathrm{prem}(f, \mathrm{CS}^{(i)}) = 0\}$, *for* $i = 1 \cdots r$. *Then for all* i, $\mathrm{PD}^{(i)}$ *is a prime ideal with an irreducible characteristic set* $\mathrm{CS}^{(i)}$. *Moreover, we have*

$$\mathcal{V}(\mathrm{CS}^{(i)}) \setminus \mathcal{V}(J^{(i)}) \subseteq \mathcal{V}(\mathrm{PD}^{(i)}) \subseteq \mathcal{V}(\mathrm{PS}^{(i)}) \subseteq \mathcal{V}(\mathrm{CS}^{(i)}) \tag{2.9}$$

and a decomposition of the corresponding algebraic set of PS

$$\mathcal{V}(\mathrm{PS}) = \bigcup_{i=1}^{r} \mathcal{V}(\mathrm{PD}^{(i)}), \tag{2.10}$$

where $\mathrm{PS}^{(i)}$ *is the corresponding polynomial set of* $\mathrm{CS}^{(i)}$ *(i.e. QS in Algorithm 2.1.4).*

We even have a constructive way to compute a finite generating set of $\mathrm{PD}^{(i)}$, for all i, from its irreducible characteristic set $\mathrm{CS}^{(i)}$ as follows.

Theorem 2.2.1 *Let* $\mathrm{CS} = f_1, \ldots, f_h$ *be an irreducible ascending chain with an extended zero* $\zeta = (u_1, \ldots, u_{k-h}, \zeta_1, \ldots, \zeta_h)$, $\mathrm{PD} = \{f : \mathrm{prem}(f, \mathrm{CS}) = 0\}$ *and* $\mathcal{C} = \mathrm{CS} \cup \{g\}$, *where* $g = y \cdot I_1 \cdots I_h - 1$, y *is a new variable and* $I_j = \mathrm{lc}(f_j)$, $j = 1 \cdots h$. *Fix the order on the variables such as* $u_1 \prec \cdots \prec u_{k-h} \prec x_1 \prec \cdots \prec x_h \prec y$. *We have*

$$\mathrm{PD} = \langle \mathcal{C} \rangle \cap K[u_1, \cdots, u_{k-h}, x_1, \cdots, x_h]$$

From this theorem, a finite generating set of PD can be computed by, for example, $\mathrm{GB}(\mathcal{C}) \cap K[u_1, \cdots, u_{k-h}, x_1, \cdots, x_h]$, where $\mathrm{GB}(\mathcal{C})$ is a Gröbner basis of $\langle \mathcal{C} \rangle$ with respect to the lexicographic ordering. In fact, we can compute a Gröbner basis with respect to a total degree ordering, which is much easier to obtain, and then transform it to the Gröbner basis with respect to the lexicographic ordering by using some means from linear algebra.

The decomposition of $\mathcal{V}(\mathrm{PS})$ in (2.10) may be contractible, i.e. some varieties may be a subvariety of the others and therefore can be removed. Suppose that we are working over an algebraically closed field, then the following theorem gives a necessary and sufficient condition for a complete removal of redundant components.

Theorem 2.2.2 *Given two irreducible ascending chains* $\mathrm{CS}^{(1)}$ *and* $\mathrm{CS}^{(2)}$ *together with their corresponding prime ideas* $\mathrm{PD}^{(1)}$ *and* $\mathrm{PD}^{(2)}$ *respectively. We have*

$$\mathcal{V}(\mathrm{PD}^{(1)}) \subseteq \mathcal{V}(\mathrm{PD}^{(2)}) \Leftrightarrow \mathrm{prem}(f, \mathrm{CS}^{(1)}) = 0$$

for all f in the finite generating set of $PD^{(2)}$.

However, computation of Gröbner bases is expensive; therefore, it would be more efficient if we can reduce the number of Gröbner basis computations by comparing the irreducible components directly just based on their ascending chains. For example, according to the affine dimension theorem in algebraic geometry, the dimension of an irreducible component of $\mathcal{V}(\mathrm{PS})$ can not be less than $k - h$, where k is the number of variables and h is the number of polynomials in PS. Therefore, if for some i, the number of polynomials in $\mathrm{CS}^{(i)}$ is greater than h then $\mathrm{PD}^{(i)}$ has to define a redundant subvariety of $\mathcal{V}(\mathrm{PS})$ and hence can be removed. Additionally, the following necessary conditions are also useful.

Corollary 2.2.1 *Let* $\mathrm{CS}^{(1)}$, $\mathrm{CS}^{(2)}$, $\mathrm{PD}^{(1)}$ *and* $\mathrm{PD}^{(2)}$ *as in Theorem 2.2.2.*

- *If* $\mathrm{prem}(f_i, \mathrm{CS}^{(1)}) = 0$ *and* $\mathrm{prem}(lc(f_i), \mathrm{CS}^{(1)}) \neq 0$ *for all f_i in* $\mathrm{CS}^{(2)}$ *then* $\mathcal{V}(\mathrm{PD}^{(1)}) \subseteq \mathcal{V}(\mathrm{PD}^{(2)})$.
- *Assume that* $\mathrm{CS}^{(1)}$ *and* $\mathrm{CS}^{(2)}$ *have the same set of dependent variables then* $\mathcal{V}(\mathrm{PD}^{(1)}) = \mathcal{V}(\mathrm{PD}^{(2)})$ *if and only if* $\mathrm{prem}(f_i, \mathrm{CS}^{(1)}) = 0$ *for all f_i in* $\mathrm{CS}^{(2)}$.

2.2.1 Elementary Properties of Dimension

The following elementary properties of dimension can be found, for example in [CLO92].

Theorem 2.2.3 *Let V_1 and V_2 be projective or affine algebraic sets over an arbitrary field K. If $V_1 \subseteq V_2$ then* $\dim V_1 \leq \dim V_2$.

We next study the relation between the dimension of an algebraic set and the number of its defining equations. We begin with the case where V is defined by a single equation.

Theorem 2.2.4 *Let K be an algebraically closed field and let $f \in K[x_0, x_1, \ldots, x_n]$ be a nonconstant homogeneous polynomial. Then the dimension of the projective algebraic set in $\mathbb{P}^n(K)$ defined by f is*

$$\dim \mathcal{V}(f) = n - 1$$

We can prove an analogous statement for affine hypersurfaces. It is important to note that these results are not valid if K is not algebraically closed. We have the following relation for the dimension of the intersection.

Theorem 2.2.5 *Let K be an algebraically closed field and I be a homogeneous ideal in $K[x_0, x_1, \ldots, x_n]$. If $f \in K[x_0, x_1, \ldots, x_n]$ is any nonconstant homogeneous polynomial, then*

$$\dim \mathcal{V}(I) \geq \dim \mathcal{V}(I + \langle f \rangle) \geq \dim \mathcal{V}(I) - 1$$

Corollary 2.2.2 *Let K be an algebraically closed field and I be a homogeneous ideal in $K[x_0, x_1, \ldots, x_n]$. If $f \in K[x_0, x_1, \ldots, x_n]$ is any nonconstant homogeneous polynomial whose class in the quotient ring $K[x_0, x_1, \ldots, x_n]/I$ is not a zero divisor, then*

$$\dim \mathcal{V}(I + \langle f \rangle) = \dim \mathcal{V}(I) - 1$$

2.2.2 Dimension of the Irreducible Components

Given an irreducible ascending chain, the difference of the algebraic set defined by the ascending chain from the algebraic set defined by the product of its initials may not be an algebraic set. But the following theorem, which is proposed by the author and first mentioned in [Tra96c], shows the Zariski closure of the difference.

Theorem 2.2.6 *Let* CS *be an irreducible ascending chain,* PD *be the corresponding prime ideal and J be the product of the initials of the polynomials in* CS. *Then* $\mathcal{V}(\mathrm{PD})$ *is the Zariski closure of* $\mathcal{V}(\mathrm{CS}) \setminus \mathcal{V}(J)$, *i.e.*

$$\mathcal{V}(\mathrm{PD}) = \overline{\mathcal{V}(\mathrm{CS}) \setminus \mathcal{V}(J)}.$$

In other words, $\mathcal{V}(\mathrm{PD})$ *is the smallest algebraic set containing* $\mathcal{V}(\mathrm{CS}) \setminus \mathcal{V}(J)$.

Proof. Assume that $\mathcal{I}$ is an ideal in $K[u_1, \cdots, u_{k-h}, x_1, \cdots, x_h]$ such that $\mathcal{V}(\mathrm{CS}) \setminus \mathcal{V}(J) \subseteq \mathcal{V}(\mathcal{I})$. Let f be an arbitrary polynomial in $\mathcal{I}$ and R be the pseudoremainder of f with respect to CS then we have $R(P) = 0$ for all $P \in \mathcal{V}(\mathrm{CS}) = (\mathcal{V}(\mathrm{CS}) \setminus \mathcal{V}(J)) \cup (\mathcal{V}(\mathrm{CS}) \cap \mathcal{V}(J))$. By Lemma 2.2.1, R vanishes in particular, at every zero of PD. Since PD is prime, R is a polynomial in PD by Hilbert's Nullstellensatz and hence $R(\zeta) = 0$ where ζ is an extended zero of CS. That means, $R \equiv 0$ and $f \in \mathrm{PD}$. Therefore, $\mathcal{V}(\mathrm{PD}) \subseteq \mathcal{V}(\mathcal{I})$. □

We now introduce the concept of pseudoirreducibility. This is a new idea of the author's which has been first reported in [Tra96a].

Definition 2.2.1 *Given an algebraic set $\mathcal{V}$, a set of affine sets $\{\widehat{V^{(1)}}, \ldots, \widehat{V^{(r)}}\}$ is called a pseudoirreducible decomposition of $\mathcal{V}$ if for all $i = 1 \ldots r$, $\widehat{V^{(i)}}$ coincides with $V^{(i)}$ except at a measure zero set of points (i.e. a set whose dimension is strictly smaller then $\dim V^{(i)}$), where $V^{(1)}, \ldots V^{(r)}$ is the irreducible decomposition of $\mathcal{V}$.*

Note that $\widehat{V^{(i)}}$ need not be an algebraic set. We will state and prove the following theorem on the pseudoirreducible decomposition.

Theorem 2.2.7 *Given a system* PS *of polynomials with a noncontractible sequence of irreducible extended characteristic sets* $\mathrm{CS}^{(1)}, \mathrm{CS}^{(2)}, \ldots, \mathrm{CS}^{(r)}$, *then*

$$(\mathcal{V}(\mathrm{CS}^{(1)}) \setminus \mathcal{V}(J^{(1)})), \ldots, (\mathcal{V}(\mathrm{CS}^{(r)}) \setminus \mathcal{V}(J^{(r)}))$$

is a pseudoirreducible decomposition of $\mathcal{V}(\mathrm{PS})$

Proof. From Ritt's theorem, we have

$$\mathcal{V}(\mathrm{PS}) = \bigcup_{i=1}^{r} (\mathcal{V}(\mathrm{CS}^{(i)}) \setminus \mathcal{V}(J^{(i)})).$$

For all $1 \le i \le r$, let $\mathrm{PD}^{(i)} = \{f : \mathrm{prem}(f, \mathrm{CS}^{(i)}) = 0\}$, then $\mathrm{PD}^{(i)}$ is a prime ideal and hence $\mathcal{V}(\mathrm{PD}^{(i)})$ is irreducible. Moreover, we have

$$(\mathcal{V}(\mathrm{CS}^{(i)}) \setminus \mathcal{V}(J^{(i)})) \subseteq \mathcal{V}(\mathrm{PD}^{(i)}) \subseteq \mathcal{V}(\mathrm{PS}^{(i)}) \subseteq \mathcal{V}(\mathrm{CS}^{(i)}).$$

From Theorem 2.2.6, $\mathcal{V}(\mathrm{PD}^{(i)})$ is the Zariski closure of $\mathcal{V}(\mathrm{CS}^{(i)}) \setminus \mathcal{V}(J^{(i)})$. We will now prove that there is only a measure zero set of points which are members of $\mathcal{V}(\mathrm{PD}^{(i)})$ but not $\mathcal{V}(\mathrm{CS}^{(i)}) \setminus \mathcal{V}(J^{(i)})$. Let P be such a point, then $P \in \mathcal{V}(\mathrm{PD}^{(i)})$ and $P \notin \mathcal{V}(\mathrm{CS}^{(i)})$ or $P \in \mathcal{V}(J^{(i)})$. But the former cannot occur, therefore, $P \in \mathcal{V}(\mathrm{PD}^{(i)} + \langle J^{(i)} \rangle) = \mathcal{V}(\mathrm{PD}^{(i)} + \langle I_1^{(i)} \cdots I_r^{(i)} \rangle)$. Homogenize the prime ideal $\mathrm{PD}^{(i)}$ obtaining $(\mathrm{PD}^{(i)})^h$ which is a homogenous prime ideal. Since $\mathrm{CS}^{(i)}$ is irreducible, for all $j = 1 \ldots r$, $I_j^{(i)} \notin \mathrm{PD}^{(i)}$ and then $J^{(i)} \notin \mathrm{PD}^{(i)}$, where $I_j^{(i)}$s are coefficients of polynomials in $\mathrm{CS}^{(i)}$. Since $K[x_0, x_1, \ldots, x_n]/(\mathrm{PD}^{(i)})^h$ is an integral domain, the class of $J^{(i)}$ is not zero in $K[x_0, x_1, \ldots, x_n]/(\mathrm{PD}^{(i)})^h$ and hence it is not a zero divisor. From Corollary 2.2.2, we have

$$\dim \mathcal{V}(\mathrm{PD}^{(i)} + \langle J^{(i)} \rangle) = \dim(\mathrm{PD}^{(i)}) - 1$$

Therefore, $\dim \Big(\mathcal{V}(\mathrm{PD}^{(i)}) \setminus (\mathcal{V}(\mathrm{CS}^{(i)}) \setminus \mathcal{V}(J^{(i)})) \Big) = \dim(\mathrm{PD}^{(i)}) - 1$. In case of a one-dimensional algebraic set, we have only finitely many points which are members of the irreducible components $\mathcal{V}(\mathrm{PD}^{(i)})$s but not members of the pseudoirreducible components $(\mathcal{V}(\mathrm{CS}^{(i)}) \setminus \mathcal{V}(J^{(i)}))$s. $\Box$

Note that we have $\dim \mathcal{V}(\mathrm{CS}^{(i)}) = \dim \mathcal{V}(\mathrm{PS}^{(i)})$ but may not have $\dim(\mathcal{V}(\mathrm{CS}^{(i)}) \setminus \mathcal{V}(\mathrm{PS}^{(i)})) < \dim(\mathcal{V}(\mathrm{CS}^{(i)}))$.

We found that the above idea of avoiding heavy computations of generating sets is very useful in Computer Aided Geometric Design and Visualization. In particular for the approximation and representation of one-dimensional algebraic sets in higher dimensional space (see [Tra95b, Tra96b]). The advantages of this approach are that:

1. We almost have the irreducible decomposition.
2. The system of equations corresponding to each pseudoirreducible component is very simple. So it is much easier to be traced or evaluated.

A method for recovering the missing points during the tracing process is also given in [Tra96b].

2.3 Real Root Isolation of Polynomials with Real Algebraic Number Coefficients

There are several methods for isolating real roots of a univariate polynomial, for example: the methods based on Sturm's theorem, Descartes' lemma, and Rolle's theorem, in that the method based on Descartes' lemma is the most efficient in practice (see [Usp48, Joh91, Tra95c]). However, this method requires that the input polynomial is square-free. Thus in order to use this method, we first need to compute the greatest square-free divisor of a polynomial $P(\alpha, y)$ where α is a real algebraic number, for instance $\sqrt{2}$, defined by its minimal polynomial together with an interval to distinguish it from its real conjugates. This computation is usually extremely time-consuming even for low-degree polynomials. Furthermore, the resulting greatest square-free divisor usually has much bigger coefficients (represented by very dense univariate polynomials) than the original nonsquare-free polynomial, blowing up the computation of the subsequent real root isolation. Fortunately, for the use of this method in Computer Aided Geometric Design and Visualization such as in Chapter 5, we prove that the real algebraic coefficient polynomials whose real roots we want to isolate are always square-free. Hence, we can avoid that extremely time-consuming computation.

Hereafter, we recall some basic definitions and theorems for real root isolation of polynomials with real algebraic number coefficients. Readers can find proofs of most of the theorems in this section in, for instance [Usp48, Loo82].

Definition 2.3.1 (Variation) *Let $\bar{a} = (a_0, \ldots, a_n)$ be a sequence of real numbers. Then the* variation *(or number of sign changes) of $\bar{a}$, denoted $V(\bar{a})$, is the number of pairs $(i, i+k)$, $k > 1$ such that:*

1. $a_i \cdot a_{i+k} < 0$.
2. $a_{i+r} = 0$ *for* $0 < r < k$.

Definition 2.3.2 (Root Bound) *Let $P(x)$ be a polynomial with complex coefficients. A* root bound *for the polynomial $P(x)$ is a real number B such that if $P(\alpha) = 0$ then $|\alpha| < B$.*

Theorem 2.3.1 (Knuth's Bound) *Let $P(x) = \sum_{i=0}^{m} a_i x^i$ be a polynomial with complex coefficients. If $B = \max_{1 \le k \le m} |a_{m-k}/a_m|^{1/k}$, then $2B$ is a root bound for $P(x)$*

Theorem 2.3.2 (Descartes' lemma) *Let $P(x) = \sum_{i=0}^{m} a_i x^i \in \mathbb{R}[x]$ and let $V(P)$ be the variation of the sequence $(a_0, \ldots, a_m)$. Then the number $Z_+(P)$ of strictly positive real roots (counted with multiplicity) of $P(x)$ does not exceed $V(P)$, and is congruent to $V(P)$ mod 2.*

A real algebraic number, for instance $\sqrt{2}$, is a real number which satisfies an integral polynomial equation. A real algebraic number α can be represented by a square-free polynomial $A(x)$ (we do not require $A(x)$ to be an irreducible polynomial) such that $A(\alpha) = 0$ and an isolating interval, I, which distinguishes α from other real roots (not only its real conjugates). The extension field of the rational numbers generated by α is denoted by $\mathbb{Q}(\alpha)$. An element, β, in $\mathbb{Q}(\alpha)$ is represented by the polynomial $B(x)$ in the residue class ring $\mathbb{Q}[x]/\langle A(x)\rangle$ such that $\beta = B(\alpha)$. If the degree of $A(x)$ is equal to m then $1, x, \ldots, x^{m-1}$ is a basis for $Q[x]/\langle A(x)\rangle$ and we can assume that the degree of $B(x)$ is less than m. Generally, this representation is not unique because we did not require $A(x)$ to be irreducible. Since we only assume that $A(x)$ is square-free, $\mathbb{Q}[x]/\langle A(x)\rangle \cong \mathbb{Q}[x]/\langle A_1(x)\rangle \times \cdots \times \mathbb{Q}[x]/\langle A_n(x)\rangle$, where $A_1(x), \ldots, A_n(x)$ are the irreducible factors of $A(x)$. Arithmetic for real algebraic numbers is performed using polynomial operations in $\mathbb{Q}[x]/\langle A(x)\rangle$. In the same manner, for polynomial evaluation and transformation, a polynomial with algebraic coefficients will be seen as a bivariate polynomial in $\mathbb{Q}(\alpha)[y]$ (by clearing the denominator we can assume that the polynomials is in $\mathbb{Z}[\alpha][y]$). Sign computation is performed as follows: If $A(\alpha) = 0$, $\alpha \in I = (a, b]$, and $\beta = B(\alpha)$, the sign of β can be computed by refining the isolating interval I until it contains no root of $B(x)$. Hence, the sign of $\beta = B(\alpha)$ is equal to the sign of $B(x)$ for any $x \in I$.

Now we describe an algorithm for isolating positive real roots of a polynomial with algebraic coefficients P. For the other roots, one has to replace y by $-y$.

Algorithm 2.3.1 ($L \leftarrow DESCARTES(A(x), I, P(\alpha, y))$)

Inputs: $A(x)$ is the defining polynomial for a real algebraic number α. I is an isolating interval for α. $P(\alpha, y) \not\equiv 0$ is a square-free polynomial in $\mathbb{Z}[\alpha, y]$, and $y \nmid P(\alpha, y)$;

Output: L is a list of isolating intervals for the real positive roots of $P(\alpha, y)$.

1. *Let* $B \geq 1$ *be a root bound of* $P(\alpha, y)$. *Put* $Q(\alpha, y) = P(\alpha, B \cdot y)$ *(all the positive roots of* Q *are in* $[0, 1]$*).*

2. *Put* $Q^*(\alpha, y) = (y+1)^m Q(\alpha, 1/(y+1))$ *(the roots of* Q^* *in* $[0, \infty)$ *correspond to the root of* Q *in* $(0, 1]$*) and compute* $Var(Q^*) = Var(A(x), I, Q^*)$*:*

   (a) *If* $Var(Q^*) = 0$ *then* Q^* *has no positive root, so the same is true for* P.

   (b) *If* $Var(Q^*) = 1$ *then* Q^* *has one positive root, so the same is true for* P.

   (c) *If* $Var(Q^*) > 1$*, test if* $Q(1/2) = 0$*, and put:*

      i. $Q_1 = 2^m Q(\alpha, y/2)$.

      ii. $Q_2 = Q_1(\alpha, y+1) = 2^m Q(\alpha, (y+1)/2)$.

      (the roots of Q_1 *in* $[0, 1]$ *correspond to the roots of* Q *in* $[0, 1/2]$ *and the roots of* Q_2 *in* $[0, 1]$ *correspond to the roots of* Q *in* $[1/2, 1]$

3. *(Recursion)* $L_1 \leftarrow$ *DESCARTES*$(A(x), I, Q_1)$;
   $L_2 \leftarrow$ *DESCARTES*$(A(x), I, Q_2)$;
   $L \leftarrow concat(L_1, L_2)$.

Termination of the algorithm is guaranteed by the following theorem:

Theorem 2.3.3 *Let* $P(x)$ *be a square-free polynomial with real coefficients of degree* m. *If* $P(x)$ *has a single root in the interval* $(0, 1)$ *and all of the remaining real and complex roots are outside the circle of radius 1 centered at* $(0, 0)$ *and* $(1, 0)$*, then* $Var((x+1)^m P(1/(x+1))) = 1$.

An algorithm based on Descartes' lemma for isolating real roots of a polynomial within a given interval can be found in [Usp48, Joh91]. In that, we have first to transform the given interval into the unit interval $(0, 1)$. The transformation, with the use of synthetic division or Horner's method, is efficient. However, the length of the coefficients of the transformed polynomial (both numerator and denominator) roughly multiplies by the degree of the polynomial thus making the algorithm impractical.

We describe an algorithm for isolating real roots of a polynomial P within a given interval J. The algorithm is heavily used in practice. Our algorithm does not use the mentioned unit transformation. In fact, it utilizes interval arithmetic. The algorithm is much more efficient than the transformation approach and Sturm's sequence method.

Algorithm 2.3.2 ($Descartes(A(x), I, P(\alpha, y), J)$)

Inputs: A(x) is the defining polynomial for a real algebraic number α. I *is an isolating interval for* α. $P(\alpha, y) \not\equiv 0$ *is a square-free polynomial in* $\mathbb{Z}[\alpha, y]$*, and* $y \nmid P(\alpha, y)$*;* J *is an open interval.*

Output: L is a list of isolating intervals for the real roots of $P(\alpha, y)$ *within the open interval* J.

1. *Let Lp be a list of isolating intervals for all of the real roots (positive and negative) of the polynomial in the real axis. We can efficiently compute Lp by using Algorithm 2.3.1.*

2. **While** $l \in Lp$,
   If $l \in J$ *then* $L \leftarrow cons(l, L)$.
   else if l *and* J *are disjointed then* **next**.
   else refine the interval l *using, say bisection method,*
   until it is contained in J *or disjointed from* J.
   end-while

The algorithms were implemented in Maple V.3. In [Tra95c], we report a considerable speed up of computing time and reductions in memory consumption of Algorithm 2.3.2 in comparison with the transformation approach and Sturm's sequence. In that, for isolating real roots of discriminants of degree 10 bivariate polynomials with 10 bits coefficients, the average speed up of the algorithm is 350 times and 2800 times respectively; the algorithm requires a fraction of 1/18 of the time and 1/65 of the memory consumption, respectively. The results were obtained on a SGI-IRIS workstation under IRIX 5.2 with 64MB RAM using MIPS-RISC 4000A/100MHz CPU.

Chapter 3

Extending Dixon's Resultants

Dixon's resultant method is an efficient way of simultaneously eliminating several variables from a system of nonhomogeneous polynomial equations at a time. However, the method only works for systems of $n+1$ generic n-degree polynomials in n variables and does not work for most algebraic and geometric problems. In this chapter, by using techniques from pseudoinverse theory, we extend Dixon's resultant to an arbitrary system of $n+1$ nontrivial polynomials where the Dixon matrix can be singular or even nonsquare. This is a new result of the author's, which has been first presented at the workshop "Automated Deduction in Geometry" in September 1996, Toulouse, France.

3.1 Introduction

In applied mathematics such as CAGD and solid modeling, we often have to eliminate several variables from a system of equations (e.g. the problem of implicitization). Theoretically, the Gröbner bases method, Ritt's characteristic set method or successive Sylvester resultant computations can be used to achieve our demands. But they are inefficient and require a lot of memory space.

Meanwhile, Dixon's resultant is an efficient method to simultaneously eliminate several variables from a system of nonhomogeneous polynomial equations at a time. In contrast with the successive resultant computation techniques which eliminate variables one by one and compute numerous intermediate resultants before finally computing the resultant of the whole set, Dixon's resultant creates a much smaller matrix and avoids such intermediate results, and hence is much faster and also saves memory space.

But unfortunately, Dixon's resultant does not work for most algebraic and geometric problems. Originally, Dixon's resultant was a method for obtaining the resultant of a set of three generic bidegree polynomials (see [Dix08]). This is a generalization of Cayley's formulation [Cay65] of Bezout's efficient method for computing the resultant of two univariate polynomials. Dixon had generalized his method to any $n+1$ generic n-degree polynomials in n variable – a condition which cannot be satisfied by most problems arising in practice. For most practical problems, the matrix set up in Dixon's method, the Dixon matrix, becomes singular or even nonsquare. As a consequence, Dixon's resultant vanishes identically, without providing any information about the common solutions of equations. There are some efforts to obtain nonzero conditions by perturbing the polynomial system (see [Chi90], [Can90]) or checking if there exists a submatrix of the Dixon matrix with one column less and strictly smaller rank than the Dixon matrix when the Dixon matrix is square and singular (see [KSY94], [KS95]). However these efforts either suffer from the facts that the method is not automatic, requires human expertise and enlarges the size of Dixon matrix with larger entries; or requires a precondition on the system of equations and lacks of a formal explanation.

In this chapter, by using techniques from pseudoinverse theory, we overcome some of the above obstacles and extend Dixon's resultant to an arbitrary system of $n+1$ nonzero polynomials where the Dixon matrix can be singular or even nonsquare. It turns out that the result in [KSY94] is a special case of our results. Note that, this approach can also be used to extend other multivariate resultant methods such as Macaulay resultant or sparse resultant.

3.2 Dixon's Resultant

Let $\mathcal{F} = \{p_1(x_1,\ldots,x_n),\ldots,p_{n+1}(x_1,\ldots,x_n)\}$ be a set of $n+1$ generic n-degree polynomials in n variables, i.e. there exist nonnegative integers m_1, m_2, $\ldots$, m_n such that $p_j = \sum_{i_1=0}^{m_1}\cdots\sum_{i_n=0}^{m_n} a_{j,i_1,\ldots,i_n} x_1^{i_1}\cdots x_n^{i_n}$, for $1 \le j \le n+1$. Let $d_{\max_i} = \max(\mathrm{degree}(p_1,x_i),\ldots,\mathrm{degree}(p_{n+1},x_i))$, for all $1 \le i \le n$. We form an $(n+1)\times(n+1)$ determinant as follows:

$$\Delta(x_1,\ldots,x_n,\alpha_1,\ldots,\alpha_n) = \begin{vmatrix} p_1(x_1,x_2,\ldots,x_n) & \cdots & p_{n+1}(x_1,x_2,\ldots,x_n) \\ p_1(\alpha_1,x_2,\ldots,x_n) & \cdots & p_{n+1}(\alpha_1,x_2,\ldots,x_n) \\ p_1(\alpha_1,\alpha_2,\ldots,x_n) & \cdots & p_{n+1}(\alpha_1,\alpha_2,\ldots,x_n) \\ \cdots & \cdots & \cdots \\ p_1(\alpha_1,\alpha_2,\ldots,\alpha_n) & \cdots & p_{n+1}(\alpha_1,\alpha_2,\ldots,\alpha_n) \end{vmatrix}$$

where $\alpha_1,\alpha_2,\ldots,\alpha_n$ are n new variables and $p_i(\alpha_1,\alpha_2,\ldots,\alpha_k,x_{k+1},\ldots,x_n)$ stands for uniformly replacing x_j by α_j for $1 \le j \le k$ in p_i. Each of $x_i = \alpha_i$, for all $1 \le i \le n$, is a zero of Δ, so they can be removed by dividing Δ by $\prod_{i=1}^{n}(x_i-\alpha_i)$. Let

$$\delta(x_1,\ldots,x_n,\alpha_1,\ldots,\alpha_n) = \frac{\Delta(x_1,\ldots,x_n,\alpha_1,\ldots,\alpha_n)}{(x_1-\alpha_1)\cdots(x_n-\alpha_n)}.$$

The polynomial δ, called the Dixon polynomial of $\mathcal{F}$, is of degree $((n+1-i)\cdot d_{\max_i})-1$ in α_i and $(i\cdot d_{\max_i})-1$ in x_i, for $1 \le i \le n$.

Any common zero of $\mathcal{F}$, say $(x_1 = c_1,\ldots,x_n = c_n)$ makes the Dixon polynomial vanish, no matter what the values of $\alpha_1,\ldots,\alpha_n$; hence all the coefficients of the various power products of $\alpha_1,\ldots,\alpha_n$ in the Dixon polynomial vanish. Let $\mathcal{E}'$ be the set of all polynomials in $x_1,\ldots,x_n$ which are coefficients of the power products of $\alpha_1,\ldots,\alpha_n$ in δ. This set then has exactly

$$\prod_{i=1}^{n}((n+1-i)\cdot d_{\max_i}) = n!\cdot\prod_{i=1}^{n} d_{\max_i} = s$$

equations (one for each power product of $\alpha_1,\ldots,\alpha_n$), each of which is of degree $(i\cdot d_{\max_i})-1$ in x_i. Also, there are

$$\prod_{i=1}^{n}(i\cdot d_{\max_i}) = n!\cdot\prod_{i=1}^{n} d_{\max_i} = s$$

power products in $x_1,\ldots,x_n$ in the equations of $\mathcal{E}'$. Let D be the $s\times s$ coefficient matrix of $\mathcal{E}'$. Then

$$\mathcal{E}' \equiv D \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_n \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \\ \vdots \\ x_n^2 \\ \vdots \\ \prod_{i=1}^{n} x_i^{i \cdot d_{\max_i} - 1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

If each power product of $x_1, \ldots, x_n$ (including $x_1^0 \cdots x_n^0 = 1$) is viewed as a new variable, say v_i, then we get a set $\mathcal{E}$ of s homogeneous linear equations in s variables

$$\mathcal{E} \equiv D \begin{pmatrix} v_1 \\ \vdots \\ v_s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

If there exists a common affine zero of $\mathcal{F}$, say $(x_1 = c_1, \ldots, x_n = c_n)$, then this also is a solution to $\mathcal{E}'$. This results in a nonzero solution to $\mathcal{E}$ i.e. $(v_1 = 1, v_2 = c_1, \ldots)$. Hence, as before, if $\mathcal{F}$ has a common affine zero, then $\mathcal{E}$ has a nonzero solution, implying that the determinant of D is zero. This gives a necessary condition on the coefficients of $p_1, \ldots, p_{n+1}$ for them to have a common zero. We call D the Dixon matrix and its determinant Dixon's resultant. It was proved by Dixon in [Dix08] that for $n+1$ generic n-degree polynomials in n variables, vanishing of Dixon's resultant is a necessary and sufficient condition for them to have a nonzero projective zero, or a necessary condition for the existence of an affine zero.

However, the method does not work for most algebraic and geometric problems. For most practical problems, the polynomials are not generic and the matrix set up in Dixon's method, the Dixon matrix, becomes singular or even nonsquare. As a consequence, Dixon's resultant vanishes identically, without providing any information about the common solutions of equations.

In the next section, by using techniques from pseudoinverse theory, we extend Dixon's resultant to an arbitrary system of $n+1$ nonzero polynomials in n variables where the Dixon matrix can be singular or even nonsquare. This is a new result of the author's. It turns out that the result in [KSY94] is a special case of our results.

3.3 Extending Dixon's Resultant

We first extend Dixon's resultant method for an arbitrary system $\mathcal{F} \subset K[x_1, \ldots, x_n]$ of $n+1$ equations in n variables, where K is a field of any characteristic set. We construct Δ as in the previous subsection, dividing its determinant by $(x_1 - \alpha_1) \cdots (x_n - \alpha_n)$ to get the Dixon polynomial δ. This polynomial may now have a degree less than or equal to $((n+1-i) \cdot d_{\max_i}) - 1$ in each α_i and less than or equal to $(i \cdot d_{\max_i}) - 1$ in each x_i for all $1 \leq i \leq n$. We construct the set of linear equation $\mathcal{E}$ from the Dixon polynomial as before, except that now $\mathcal{E}$ may have less than or equal to s equations in less than or equal to s variables. We continue to call the coefficient matrix of $\mathcal{E}$ the Dixon matrix D of size $s_1 \times s_2$ and $v = (v_1, \ldots, v_{s_2})$ the s_2-tuple of the power products of $x_1, \ldots, x_n$ which are viewed as new variables. Note that the Dixon matrix now may be singular or even nonsquare.

Definition 3.3.1 *Let $\mathcal{F} \subset K[x_1, \ldots, x_n]$ be a system of $n+1$ equations in n variables. We call $(b_1, \ldots, b_n) \in K^n$ a* nontrivial common root *of $\mathcal{F}$ if*

$$(v_1(b_1, \ldots, b_n), \ldots, v_{s_2}(b_1, \ldots, b_n)) \neq \vec{0}$$

Obviously, a common root of $\mathcal{F}$ in the algebraic torus $(K \setminus \{0\})^n$ is a nontrivial common root.

Theorem 3.3.1 *Given a system $\mathcal{F} \subset K[x_1, \ldots, x_n]$ of $n+1$ equations in n variables, D has a nonzero root if and only if*

$$\det(D^+ D) = 0,$$

where D^+ is the pseudoinverse of D of size $s_2 \times s_1$.

Proof. Assume that $D\bar{x} = 0$ then $D^+ D\bar{x} = 0$ where $\bar{x}$ is a nonzero root of D. Hence $\det(D^+ D) = 0$. Conversely, if $\det(D^+ D) = 0$ then there exists a nonzero root $\bar{x}$ of $D^+ D$. Therefore $D\bar{x} = DD^+ D\bar{x} = 0$. □

Since every matrix in $K^{s_1 \times s_2}$ has its unique pseudoinverse (see Theorem 1.1.1) and we can even obtain the pseudoinverse in an algorithmic manner (see Section 1.1.3), the theorem means that we have a necessary and sufficient condition for the existence of a nonzero root of the system $\mathcal{E}$.

Corollary 3.3.1 *Let $\mathcal{F} \subset K[x_1, \ldots, x_n]$ be a system of $n+1$ equations in n variables. If $\mathcal{F}$ has a nontrivial common root then*

$$\det(D^+ D) = 0.$$

Proof. Straightforward from Definition 3.3.1 and Theorem 3.3.1. □

We now extend the coefficient field of the polynomials to a polynomial ring $K[y_1, \ldots, y_m]$ where K is a field of any characteristic and the new indeterminates $y_1, \ldots, y_m$ are adjoined to the field K. Any ring homomorphism $\varphi : K[y_1, \ldots, y_m] \to K$ such that $y_i \mapsto a_i$ for some $a_i \in K$, $1 \leq i \leq m$, is called a specialization mapping. We also call $\varphi(D) = (\varphi(d_{ij}))$ a specialization of a Dixon matrix $D = (d_{ij})$. Note that, from now on, any affine zero of $\mathcal{F}$, if it exists, is not a solution to $\mathcal{E}$ but one of its specialization.

Lemma 3.3.1 *Let D, F, G be matrices of appropriate sizes with entries being in K; a, b, c be in K; and φ be a specialization mapping. Then we have:*

1. $\varphi(a \cdot F \cdot G) = a \cdot \varphi(F) \cdot \varphi(G)$,
2. $\varphi(b \cdot F + c \cdot G) = b \cdot \varphi(F) + c \cdot \varphi(G)$,
3. $\det(\varphi(D)) = \varphi(\det(D))$, *and* $\mathrm{rank}\varphi(D) \leq \mathrm{rank}D$,
4. $(\varphi(F))^+ = \varphi(F^+)$.

Proof. (1), (2) and (3) are obvious. For proving (4), it is easy to check that $\varphi(F^+)$ satisfies Equation (1.1)–(1.4). □

Let $\mathcal{F} \subset K[y_1, \ldots, y_m][x_1, \ldots, x_n]$ be a nonempty set of $n+1$ equations in n variables $x_1, \ldots, x_n$. For any m-tuple $(a_1, \ldots, a_m) \in K^m$, we denote the corresponding specialization mapping defined by the tuple φ_a.

Definition 3.3.2 *Let $\mathcal{F} \subset K[y_1, \ldots, y_m][x_1, \ldots, x_n]$ be a system of $n+1$ equations in n variables. We call $(a_1, \ldots, a_m, b_1, \ldots, b_n)$ a* nontrivial common root *of $\mathcal{F}$ if*

$$(v_1(b_1, \ldots, b_n), \ldots, v_{s_2}(b_1, \ldots, b_n)) \neq \vec{0},$$

i.e. $(b_1, \ldots, b_n)$ is a nontrivial common root of $\varphi_a(\mathcal{F})$.

Lemma 3.3.2 *Let $(a_1, \ldots, a_m)$ be an m-tuple in K^m, $\mathcal{F} \subset K[y_1, \ldots, y_m][x_1, \ldots, x_n]$ be a nonempty set of $n+1$ equations in n variables $x_1, \ldots, x_n$. $\varphi_a(D)$ has a nonzero root if and only if*

$$\varphi_a(\det(D^+D)) = 0,$$

Proof. Let w be a nonzero root of $\varphi_a(D)$. Then $(\varphi_a(D))^+\varphi_a(D)w = \varphi_a(D^+D)w = 0$ and hence $\varphi_a(\det(D^+D)) = \det(\varphi_a(D^+D)) = 0$, where $\det(D^+D)$ is a polynomial in $K[y_1, \ldots, y_m]$. Conversely, if $\varphi_a(\det(D^+D)) = 0$ then there exists a nonzero affine zero w of $\varphi_a(D))^+\varphi_a(D)$. Therefore, $\varphi_a(D)\varphi_a(D))^+\varphi_a(D)w = \varphi_a(D)w = 0$ □

Lemma 3.3.3 *Given a system $\mathcal{F} \subset K[y_1, \ldots, y_m][x_1, \ldots, x_n]$ of $n+1$ equations in n variables. If $D \in K(y_1, \ldots, y_m)_{s_2}^{s_1 \times s_2}$ is of full column rank then there exists a* nontrivial necessary condition *for the fact that the system $\mathcal{F}$ has a nontrivial common root. Moreover, for any square submatrix D' of the same rank with D,* $\det(D') = 0$ *is such a necessary condition.*

Proof. From the previous lemma, $\det(D^+D) = 0$ is such a necessary condition. Let (a, b) be a nontrivial common root of $\mathcal{F}$, D' be any nonsingular submatrix of D. Then $\mathrm{rank}\varphi_a(D) = \mathrm{rank}\varphi_a(D') < \mathrm{rank}D$ and hence $\varphi_a(D')$ must be singular. Therefore $\varphi_a(\det(D')) = 0$ where $\det(D')$ is not identical to zero. □

In case the Dixon matrix is not of full column rank, we need to use the full-rank factorization technique in Section 1.1. From Lemma 1.1.4, for any matrix D, there exist matrices F and G such that

- $D = FG$,
- F is of full-column rank,
- G is of full-row rank.

In that, we can choose F be any matrix whose columns are a basis for the range space of D. Then the matrix G is uniquely determined. Alternately, we can choose G be any matrix whose rows are a basis for the row space of D. The matrix G can be obtained by, for example, performing elementary row operations on the matrix D. Then the matrix F is uniquely determined. Actually, the full-rank factorization is very useful for improving our method as we will see in the next lemmas.

Lemma 3.3.4 *If* $\mathrm{rank}\varphi_a(D) < \mathrm{rank}D$ *for a common root (a, b) of $\mathcal{F}$ then there exists a nontrivially necessary condition. Moreover, for any square submatrix D' of the same rank with D,* $\det(D') = 0$ *is such a necessary condition.*

Proof. We can extend D' to G, a matrix whose rows are a basis for the row space of D. Let $D = FG$ be the corresponding full-rank factorization of D for an appropriate matrix F. Then $\mathrm{rank}\varphi_a(G) < \mathrm{rank}G$ and hence $\varphi_a(\det(D')) = 0$ □

Lemma 3.3.5 *Let $D = FG$ is a full-rank factorization of D. If* $\mathrm{rank}\varphi(D) = \mathrm{rank}D$ *for all common roots of $\mathcal{F}$ then $\varphi(D) = \varphi(F)\varphi(G)$ is a full-rank factorization of $\varphi(D)$.*

PROOF. Otherwise, rank$\varphi(D)$ would less than rank$D$. $\square$

The following corollary is free of any specialization.

Corollary 3.3.2 *If there exists a submatrix $\bar{D}$ of size $s_1 \times (s_2 - 1)$ by deleting the i^{th} column of D such that* rank$\bar{D}$ $<$ rank$D$ *then there exists a nontrivial necessary condition for the fact that the system $\mathcal{F}$ has a nontrivial common root (a, b) where $v_i(b) \neq 0$. Moreover, for any square submatrix D' of the same rank with D,* $\det(D') = 0$ *is such a necessary condition.*

PROOF. Since $v_i(b) \neq 0$, we have $\varphi_a\text{rank}\bar{D} = \varphi_a\text{rank}D < \text{rank}D$. The assertion comes directly from Lemma 3.3.4. $\square$

In some applications such as implicitization and Geometric Reasoning, we often know in advance that there exists a common root of the system $\mathcal{F}$ in the algebraic torus $(K \setminus \{0\})^n$. In this case, the following corollary is straightforward but found very useful.

Corollary 3.3.3 *Let $\mathcal{F}$ be a system of $n + 1$ equations in n variable. If D is of full row rank then there exists a nontrivial necessary condition for the fact that the system $\mathcal{F}$ has a nontrivial common root (a, b) in the torus $(K \setminus \{0\})^n$. Moreover, for any square submatrix D' of the same rank with D,* $\det(D') = 0$ *is such a necessary condition.*

From the above lemmas, we could find a nontrivial necessary condition if we could get a complete overview of the solutions of $Dx = 0$. Fortunately, this can be achieved by computing the null space of D. Let $D = FG$ be a full-rank factorization of D. The null space of G is also the null space of D. As in Chapter 1, we know in advance that the most general solution of $Gx = 0$ is $(I - G^+G)y$ where y is arbitrary. We can even find a basis for the null space of D as follows. Let the columns of G be rearranged, if necessary, so that the first r columns are linearly independent where r is the rank of D. A rearrangement of columns may be interpreted as postmultiplication by a suitable permutation matrix; thus,

$$GQ = [G_1 \;\; G_2] \quad \text{or} \quad G = [G_1 \;\; G_2]Q^T, \tag{3.1}$$

where Q is an $s_2 \times s_2$ permutation matrix (hence $Q^{-1} = Q^T$) and G_1 consists of r linearly independent columns, so that G_1 is nonsingular. Corresponding to (3.1), let the vector x in K^{s_2} be partitioned

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad x_1 \in K^r \tag{3.2}$$

In (3.1), the columns of G_2 are linear combinations of the columns of G_1, say,

$$G_2 = G_1T \quad \text{or} \quad T = G_1^{-1}G_2 \in K^{r\times(s_2-r)}, \tag{3.3}$$

where the matrix T is called the *multiplier* corresponding to the partition (3.1). Using (3.1), (3.2) and (3.3) we have

$$Gx = G_1[I_r \;\; T]Q^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0, \tag{3.4}$$

whose general solution is

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = Q \begin{bmatrix} -T \\ I_{s_2-r} \end{bmatrix} y, \tag{3.5}$$

where $y \in K^{s_2-r}$ is arbitrary. Note that the null space of G is completely determined by the multiplier T and the permutation matrix Q. In fact (3.5) shows that the columns of the $s_2 \times (s_2 - r)$ matrix

$$Q \begin{bmatrix} -T \\ I_{s_2-r} \end{bmatrix}$$

form a basis for the null space of G and therefore a basis for the null space of D.

3.4 Main Algorithm

In this section, we elaborate the main algorithm.

Algorithm 3.4.1 (Dixon)

Inputs: A system of $n+1$ nonzero polynomials in n variables.

Output: A nontrivial necessary condition for the fact that the system $\mathcal{F}$ has a nontrivial common root.

1. *Setup the Dixon matrix D of $\mathcal{F}$;*
2. **If** *D is of full column rank* **then**
   (a) *Compute a square submatrix D' of the same rank with D by using, for example Bareiss' Gaussian Elimination (see [Bar68]).*
   (b) **Return***$(\det(D'))$.*
3. **If** *D is of full row rank and we know in advance that there exists a nontrivial common root in the algebraic torus $(K \setminus \{0\})^n$* **then**
   (a) *Compute a square submatrix D' of the same rank with D by using, for example Bareiss' Gaussian Elimination (see [Bar68]).*
   (b) **Return***$(\det(D'))$.*
4. *Compute a full-rank factorization $D = FG$.*
5. *Check the null space of G which is also the null space of D. As in Chapter 1, we know in advance that the most general solution of $Gx = 0$ is $(I - G^+G)y$ where y is arbitrary.*
6. **If** *there exists an independent column of D* **then**
   (a) *Compute a square submatrix D' of the same rank with D by using, for example Bareiss' Gaussian Elimination (see [Bar68]).*
   (b) **Return***$(\det(D'))$.*
7. **Else Return***("failure").*

REMARK: I would like to emphasize that the growth of the entries of the Dixon matrix will crucially influence the efficiency of the algorithm. If we use the classical fraction-free Gaussian elimination method then the entries of the rows computed at step i will be polynomials of degree 2^i in the indeterminates. Hence, if we specialize the indeterminates to integers of norm d then entries of norm d^{2^i} are expected after the i-th step.

That means the elements of the intermediate matrices grow considerably. Of course, we could always divide the elements in row i by their gcd. But this involves a considerable number of gcd computation.

Meanwhile, the method of Bareiss (see [Bar68]) provides guaranteed factors without having to do additional operations. In fact, intrinsic factors appear systematically during the triangularization and can be divided out in order to slow down the growth of the entries. By systematically dividing out these factors, the entries after step i will have degree $i+1$, hence when specializing to integers of norm d, only a growth of order d^{i+1} can be expected. In the next section, we discuss some examples.

3.5 Experiments

We have implemented the algorithms using the kernel of the Maple language. All the experiments were carried out on a Sun Ultra-1 workstation under SunOS 5.5 with 128MB RAM.

Example 3.5.1 (Implicitization) *Given an algebraic surface in parametric form*

$$x_1 = \frac{y_1^2 - y_2^2}{y_2}, \quad x_2 = \frac{y_1^2 - y_2^2}{y_1}, \quad x_3 = \frac{1}{y_1 \cdot y_2}.$$

The objective is to find an implicit form for the surface. We first transform the system of equations into

$$\begin{aligned} q_1 &= x_1 \cdot y_2 - (y_1^2 - y_2^2), \\ q_2 &= x_2 \cdot y_1 - (y_1^2 - y_2^2), \\ q_3 &= x_3 \cdot y_2 - 1, \\ q_4 &= t \cdot y_1 \cdot y_2 - 1. \end{aligned}$$

The objective now is to eliminate t, y_2 and y_1 without any constraints on them.

We first compute the Dixon matrix which turns out to be a matrix of size 4×4 and of full column rank. Therefore, by the Lemma 3.3.3 for any square submatrix D' of the same rank with D, $\det(D') = 0$ is a necessary condition. We perform a Gaussian elimination process, compute the determinant and remove the extra factors getting an implicit form for the surface

$$x_2^4 - x_3 \cdot x_2^3 \cdot x_1^3 - 2 \cdot x_2^2 \cdot x_1^2 + x_1^4.$$

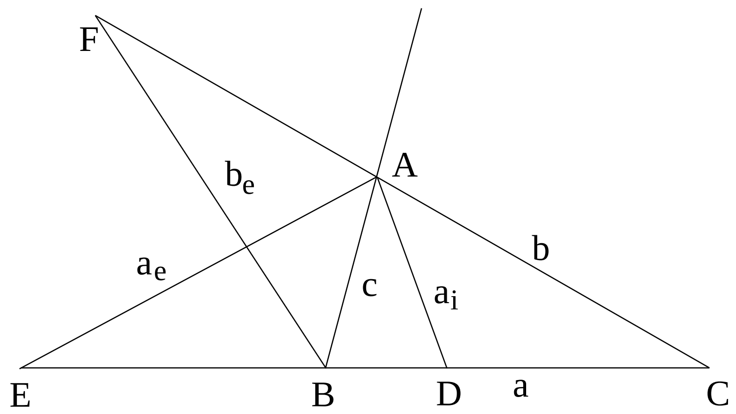
The total time taken was 0.080 second. The total memory space requirements were 298KB.

Example 3.5.2 (Elimination) *Let ABC be a triangle, a, b and c be the length of the sides BC, AC and AB, a_i and a_e be the length of internal (AD) and external (AE) bisectors of angle A, and b_e be the length of the external angle bisector (BF) of angle B.*

The objective is to determine if, given general value of the three angle bisectors, can one draw the triangle using only a compass and a ruler? As it has been proved in [Her75] that this is possible if and only if we can express a in terms of a_i, a_e and b_e, and the expression is of degree 2^m in a for some integer m. The problem can be transformed into an algebraic form as follows.

$$\begin{aligned} q_1 &= a_i^2 \cdot (b+c)^2 - c \cdot b \cdot (c+b-a) \cdot (c+b+a), \\ q_2 &= a_e^2 \cdot (c-b)^2 - c \cdot b \cdot (a+b-c) \cdot (c-b+a), \\ q_3 &= b_e^2 \cdot (c-a)^2 - a \cdot c \cdot (a+b-c) \cdot (c+b-a). \end{aligned}$$

The objective now is to eliminate b and c without any constraints on them. This example has been used in [GW92] and [KSY94]. In that for solving the problem they



took about 19 hours on their implementation in Lisp on a SUN 4 workstation and 501 seconds on their implementation in Maple on a SUN sparc 10 respectively.

We first compute the Dixon matrix which turns out to be a matrix of size 13×14 *and of full row rank. Since we implicitly assume that* b *and* c *are not zero, by the Corollary 3.3.3 for any square submatrix* D' *of the same rank with* D, $\det(D') = 0$ *is a necessary condition. We perform a Gaussian elimination process, compute the determinant and remove the extra factors getting an expression in* a, a_i, a_e *and* b_e *of 330 terms and of degree 20 in* a. *That means given* a_i, a_e *and* b_e, *the triangle cannot be constructed using only a compass and a ruler. The total time taken was 102.610 second. The total memory space requirements were 318,337KB.*

We present some more examples. In Table 3.1 we report the computing times (in second) and the memory consumptions (in KB) of Dixon's resultant method and two other methods, namely the Gröbner Bases method and the successive resultant method. A (†) indicates that the program went on for more than 150,000 seconds and consumed more than 70,000,000KB. A (‡) indicates that the object was too large.

Example 3.5.3 (Implicitization) *Find an implicit form for the surface given in parametric form*

$$x_1 = \frac{y_1^2 - y_2^2}{y_2}, \quad x_2 = \frac{y_1^2 - y_2^2}{y_1}, \quad x_3 = \frac{1}{y_1 - y_2}.$$

Example 3.5.4 (Implicitization) *Find an implicit form for the surface given in parametric form*

$$x_1 = \frac{y_2 + 2y_1^4 - 1}{y_2 - y_1 - 2}, \quad x_2 = \frac{y_1^2 y_2 + 1}{y_1}, \quad x_3 = \frac{1}{y_1 \cdot y_2}.$$

Example 3.5.5 (Elimination) *Eliminate* u, v *and* w *from the following system of equations.*

$$\begin{aligned} q_1 &= (x-u)^2 + (y-v)^2 - 1, \\ q_2 &= v^2 - u^3, \\ q_3 &= 2 \cdot v \cdot (x-u) + 3 \cdot u^2 \cdot (y-v), \\ q_4 &= (3w \cdot u^2 - 1) \cdot (2w \cdot v - 1) \end{aligned}$$

Example 3.5.6 (Elimination) *Determine if the following system has a common root.*

$$\begin{aligned} q_1 &:= x_2^2 + x_1^2 - 1, \\ q_2 &:= x_1 \cdot x_2 - 1, \\ q_3 &:= x_2^2 - x_1. \end{aligned}$$

Figure 3.1: Comparison of Three Methods

Example	Dixon's resultant		Gröbner Basis		successive resultant	
	time	mem.	time	mem.	time	mem.
E.g. 1	0.08	298	8.91	17600	‡	‡
E.g. 2	102.60	318300	†	†	‡	‡
E.g. 3	0.10	422	23.1	48000	‡	‡
E.g. 4	0.32	1136	†	†	‡	‡
E.g. 5	116.50	341000	†	†	‡	‡
E.g. 6	0.03	161	0.04	176	0.03	134

Chapter 4

A Method for Finding the Roots of an Arbitrary System of Equations

In this chapter, we present a new technique for determining the roots of an arbitrary system of equations iteratively, where the equations can be nonlinear algebraic or transcendental, the number of equations of the system can be greater than, less than or equal to the number of variables. In case the number of equations is equal to the number of variables, the method is similar to Newton's method. This method is a new result of the author's which has been first presented at the 12^{th} IASTED International Conference on Applied Mathematics in September 1994, Annecy, France.

The method is a hybrid symbolic-numerical method, in that we use symbolic techniques to transform a system of algebraic equations into a regular form whenever the system has some redundant parts or does not have a full-rank Jacobian matrix.

The method has a wide range of applicability. It is especially useful for applications in which we need to find some particular roots from a nonzero dimensional system of equations, i.e. a system, whose set of zeroes is nonzero dimensional.

We consider some conditions for convergence of our method. We claim that the convergence of the method is still quadratic. The method has been implemented in *Maple* and *Mathematica*. Several examples are presented to show that the method works nicely.

4.1 Introduction

Newton's method is a well-known iterative method for determining the roots of an equation

$$f(x) = 0, \tag{4.1}$$

where $f(x)$ and x are vectors of the same dimension. The iteration can be written in the form $x_{n+1} = g(x_n)$, where

$$g(x) = x - [f'(x)]^{-1} \cdot f(x). \tag{4.2}$$

When the dimension of the vectors is greater than 1, $[f'(x)]^{-1}$ is the inverse of the Jacobian matrix of f at x.

When $f(x)$ and x have different dimensions, Newton's method cannot be applied because the Jacobian matrix is a nonsquare matrix (see [IK66, Atk78]). We extend

Newton's method such that the extended method can be applied to these cases as well. The key ideas of our method are: (1) We will compute what we call the half-inverse of the Jacobian matrix instead of the inverse, which does not exist because the matrix is nonsquare. This computation only requires that the Jacobian matrix has full rank. (2) When a system of equations does not have a full-rank Jacobian matrix, we will use symbolic techniques to transform it into an underdetermined and full-rank-Jacobian-matrix system. We call such system a regular one. As we will point out later in Section 4.2.1, we can always carry out the regularization process when the equations are algebraic.

The extended method has a wide range of applicability, such as finding the roots of a system of nonlinear algebraic or transcendental equations, optimization, and so on. It is especially useful for applications in which we need to find some particular roots from a nonzero dimensional system of equations; for instance, surface-to-surface intersection, one of the most difficult problems in Computer Aided Geometric Design. By nonzero dimensional system of equations, we mean a system whose set of zeroes is nonzero dimensional.

The convergence of this iterative process is guaranteed whenever the mapping $g(x)$ maps a closed and bounded set $\mathcal{S} \subset \mathbb{C}^k$ into itself and $g(x)$ is *contracting*, i.e. $\exists\lambda < 1,\ \forall x, y \in \mathcal{S} :\ \|g(x) - g(y)\| \leq \lambda\|x - y\|$ for some norm $\|\cdot\|$ (see [PM91]), where λ is called a "Lipschitz" constant. This means that $g(x)$ has a unique *fixed point* $\alpha \in \mathcal{S}$ satisfying

$$\alpha = g(\alpha).$$

Furthermore, in case the solution is a root with multiplicity 1, the method has quadratic convergence. That means there exists a constant C such that for all n

$$\|e_{n+1}\| \leq C \cdot \|e_n\|^2$$

where $e_n = x_n - \alpha$.

4.2 Extending Newton's Method

Let $x = [x_1, x_2, \ldots, x_k]^T$ be a k-dimensional column vector with components x_1, x_2, $\ldots$, x_k in $\mathbb{C}$ and $f(x) = [f_1(x), \ldots, f_h(x)]^T$ an h-dimensional vector valued function, i.e. a column vector with components $f_1(x), \ldots, f_h(x)$ in $\mathbb{C}$. To deal with the norm of a vector, we can choose any one of the norms, for example: $\|x\|_\infty \equiv \max_{1\leq i\leq k}|x_i|$, $\|x\|_1 \equiv \sum_{i=1}^k |x_i|$ or $\|x\|_p \equiv \sqrt[p]{\sum_{i=1}^k |x_i|^p}$, where $p > 1$. We define the Jacobian matrix of f with respect to x to be the $h \times k$ matrix:

$$J_f(x) \equiv J(x) \equiv \left(\frac{\partial f_i(x)}{\partial x_j}\right), \quad i = 1 \ldots h,\ j = 1 \ldots k. \tag{4.3}$$

When $h = k$ and with the assumption that $\det|J(x)| \neq 0$ for x in a closed and bounded set $\mathcal{S}$, we will denote the inverse Jacobian matrix of f at x by $J_f^{-1}(x)$. In this case, Newton's method is a well-known iterative method for determining the roots of the equation $f(x) = 0$. The iteration can be written in the form $x_{n+1} = g(x_n)$, where

$$g(x) = x - J_f^{-1}(x) \cdot f(x). \tag{4.4}$$

Unfortunately, when $h \neq k$, the Jacobian matrix of f is not a square matrix and therefore cannot be inverted. Hence, Newton's method fails to apply to such cases. We extend Newton's method such that it will be applicable to these cases as well.

To get around the problem of a nonsquare matrix, we will use what we call the half-inverse of the Jacobian matrix. By half-inverse we mean a special form of pseudoinverses, namely full-rank pseudoinverse (see Section 1.1 for more detail about the theory of pseudoinverses). There are several techniques for obtaining the pseudoinverse of a numerical matrix; for example, Hestenes' technique or Greville's technique (see [BO71]). However, those techniques are purely numerical techniques, i.e. they can only be applied to each individual numerical matrix, and we have to take into account the problems of rounding error, word length, precision of arithmetic, and so on. Moreover, when we use such techniques in our iterative method for determining the roots of a system of equations, we have to apply the approximate process for obtaining the pseudoinverse matrix J^+ at each step of the iteration. That means, it is time consuming and has accumulated errors.

To avoid such problems, we try to find a structure formula and use symbolic computation in an effort to speed up our method. We next define half-inverse matrices and show two structure formulas where the first one is the well-known Penrose formula.

Definition 4.2.1 *The pseudoinverse of a matrix A is called a* half-inverse *of A, denoted by $A^{\pm}$, if $A \cdot A^{\pm} = I$ or $A^{\pm} \cdot A = I$.*

Note that if a matrix A has a half-inverse then it is unique.

Lemma 4.2.1 *Let $f(x)$ be a system of equations and J be the Jacobian matrix of f. For all x, if $J \cdot J^*(x)$ is nonsingular then $J^{\pm}(x) = J^* \cdot (J \cdot J^*)^{-1}(x)$ is the half-inverse of matrix $J(x)$; if $J^* \cdot J(x)$ is nonsingular then $J^{\pm}(x) = (J^* \cdot J)^{-1} \cdot J^*(x)$ is the half-inverse of matrix $J(x)$.*

PROOF. The matrix $J^{\pm}(x) = J^* \cdot (J \cdot J^*)^{-1}(x)$ is defined whenever $J(x) \cdot J^*(x)$ is nonsingular. It is easy to check that $J^{\pm}(x)$ satisfies Definition 1.1.2 and $J(x) \cdot J^{\pm}(x) = I$. The second assertion can be proved in the same manner. □

Corollary 4.2.1 *Let $f(x)$ be a system of equations and J be the Jacobian matrix of f. If $J(x)$ is of full row rank then $J^{\pm}(x) = J^* \cdot (J \cdot J^*)^{-1}(x)$ is the half-inverse of $J(x)$. If $J(x)$ is of full column rank then $J^{\pm}(x) = (J^* \cdot J)^{-1} \cdot J^*(x)$ is the half-inverse of $J(x)$.*

PROOF. Let $\{\vec{u_1}, \vec{u_2}, \ldots, \vec{u_h}\}$ be a basis of $\mathbb{C}^h$. The identity $B^*.J.J^*.B = (J^*.B)^*.(J^*.B)$ along with the fact that $A^*.A = 0$ implies $A = 0$ gives us the following result: If $J.J^*.B = 0$ then $J^*.B = 0$.
Suppose that $r_1.J.J^*.\vec{u_1} + \ldots + r_h.J.J^*.\vec{u_h} = 0$ for some $r_1, \ldots, r_h \in \mathbb{C}$. We will have $J^*.(r_1.\vec{u_1} + \ldots + r_h.\vec{u_h}) = 0$ since $J.J^*.(r_1.\vec{u_1} + \ldots + r_h.\vec{u_h}) = 0$. But $\mathrm{rank}(J^*) = \mathrm{rank}(J) = h$, so we have $r_1.\vec{u_1} + \ldots + r_h.\vec{u_h} = 0$ and hence, $r_1 = \ldots = r_h = 0$. As a result, $J.J^*.\vec{u_1}, \ldots, J.J^*.\vec{u_h}$ are linearly independent. That means, $\mathrm{rank}(J.J^*) = h$. Therefore, $J^{\pm}(x) = J^* \cdot (J \cdot J^*)^{-1}(x)$ is the half-inverse of matrix $J(x)$. The second assertion can be proved in the same manner with J and J^* to be exchanged. □

$J^{\pm}$ is a matrix of order $k \times h$. Hence, we can simulate Newton's method with the iterative function g being defined as:

$$g(x) = x - J^{\pm} \cdot f(x) \tag{4.5}$$

where $g(x)$ and x are vectors of the same dimension k.

REMARK: There were efforts to extend Newton's method by not using the pseudo inverse but a generalized inverse of the Jacobian matrix, (i.e. a matrix satisfying only Equation 1.1 and 1.2) dated back as early as 1927 with the works of T.H. Hildebrandt and later in 1970s with the works of A. Ben-Israel, M. Altman, etc. (see [BIG74]). However, their methods do not give us the roots of the system.

4.2.1 Regularization

When the system of equations is not underdetermined or does not have a full-rank Jacobian matrix then the half-inverse may not exist and we may not get a root of the system. In this section, we will use a well-known symbolic method, namely Ritt's characteristic set method, to transform an irregular system of equations into a regular one when the equations are algebraic. The characteristic set method was completely developed by J.F. Ritt in the early 1930s (see [Rit32], [Rit50]) and has been successfully used by W.T. Wu and his followers for mechanical theorem proving in geometry (see [Wu84], [Wu94], [Cho88]). Since there are a lot of different definitions of characteristic sets in the literature, the definitions used in this section are given in Section 2.1.

Definition 4.2.2 *A system of equations is called a* regular system *if it is not overdetermined and its Jacobian matrix has full rank everywhere.*

We first start with a finite set PS of polynomials and then regularize it to obtain a regular system CS, which may have a bigger set of zeroes. However, the zero set of the regularized system CS is bounded by Relation (2.3) and Relation (2.4). In fact, as we will point out later by Theorem 4.2.1 and Theorem 4.4.3, we do not need to worry about the zeroes of the initials because the extended Newton's method computes only zeroes of the closure of $\mathcal{V}(\mathrm{CS}) \setminus \mathcal{V}(J)$, i.e. roots of the original system PS.

If we apply Algorithm 2.1.2 recursively to the system $\mathrm{PS}_i \cup \mathrm{CS}$ where $\mathcal{V}(\mathrm{PS}_i) = \mathcal{V}(\mathrm{PS}_i \cup \mathrm{CS})$ and the rank of the extended characteristic set of PS_i is lower than that of CS, we would obtain, in a finite number of steps, a sequence of extended characteristic sets CS_i where $\mathcal{V}(\mathrm{PS}) = \bigcup_i (\mathcal{V}(\mathrm{CS}_i) \setminus \mathcal{V}(_i))$. The following depth-first algorithm can be used to compute an extended characteristic set CS_i with constant initials, from a system of polynomials. In this case, we have $\mathcal{V}(\mathrm{CS}_i) \subseteq \mathcal{V}(\mathrm{PS})$

Algorithm 4.2.1 (Constant-Initials)

Input: A system PS *of multivariate polynomials.*

Output: An extended characteristic set CS *of* PS *with constant initials, when it exists.*

1. $\mathcal{P} := [\mathrm{PS}]$; $\mathcal{T} := [\,]$; $\mathcal{S} := \emptyset$;
2. **while** $\mathcal{P} \neq [\,]$ **do**
   (a) QS := *last-element*($\mathcal{P}$);
   (b) $\mathcal{P}$:= *rest*($\mathcal{P}$);
   (c) CS := **E-Char-Set**(QS);
   (d) **if** CS = (1) *then* **next**;
   (e) **else**
      i. $\mathcal{S} := \mathcal{S} \cup \{\mathrm{CS}\}$;
      ii. **if** *all of the initials of* CS *are constants then*
         Return(CS).

iii. **else**
$\mathcal{P} := concat(\mathcal{P}\ ,\ [\mathrm{QS} \cup \mathrm{CS} \cup \{I\} : I = \mathrm{lc}(f), f \in \mathrm{CS}, \mathrm{class}(I) > 0]);$

3. **Return($\mathcal{S}$)**.

The algorithm does not need to compute all leaves of the tree of decomposition. In fact, it terminates whenever an extended characteristic set, which has the initials to be constants, is found. The following lemma on the termination of the algorithm is easy to prove.

Lemma 4.2.2 *If an extended characteristic set* CS_i *in the tree of decomposition is minimal then all of its initials are constants.*

Extensive computational experience shows that ascending chains in Ritt's sense or Wu's sense (i.e., for a triangular form to become an ascending chain, we do not need the condition that f_i is reduced with respect to f_j but its initial, for $j < i$) are very expensive to compute both from space and time considerations. In [CG90], Chou suggests using ascending chains in a weak sense which is a triangular form with $\mathrm{prem}(\mathrm{lc}(f_j); f_1, \ldots f_h) \neq 0$ for $j = 1, \ldots, h$. From my own experience, for a lot of practical problems, we can use the following algorithm to regularize an irregular system. The proof of correctness and termination of the algorithm can be found in [Tra92].

Algorithm 4.2.2 (Triangular-Form)

Input: A system PS *of polynomials in* $K[\tilde{x}]$

Output: A triangular form TR *of* PS.

1. $T := \mathrm{PS}$;
$\mathrm{TR}[i]$:= *the smallest polynomial in class i, for* $i = 1 \ldots k$. *(*$\mathrm{TR}[i] := 0$ *if* PS *has no polynomial of class i).*

2. **repeat**
$P := \emptyset$;
for all $g \in T$
$r = \mathrm{prem}(g, \mathrm{TR})$;
if $r \neq 0$ *then*
if $class(r) = 0$ *then* **Return(1)**.
else
if $r \prec \mathrm{TR}[\mathrm{class}(r)]$ *then* $\mathrm{TR}[\mathrm{class}(r)] = r$.
$P := P \cup \{r\}$;
$T := T \cup P$;
until $P = \emptyset$.

3. **Return**($\mathrm{TR} \setminus \{0\}$)

The following theorem serves as a bridge between regularization and our method for finding the roots of an arbitrary system of equations.

Theorem 4.2.1 (Regularization Theorem)
Given an irregular system PS *of algebraic equations. Let* $\mathrm{TR} = f_1, \ldots, f_h$ *be a triangular form of* PS *obtained by Algorithm 4.2.2. Then* TR *is not an overdetermined system. Moreover, there are infinitely many points in the affine space* $A^k(K)$ *(suppose that K has infinitely many elements) such that the Jacobian matrix of* TR *evaluated at those points has full rank.*

PROOF. By applying variable substitution $\{x_k \leftarrow y_1, \ldots, x_1 \leftarrow y_k\}$ to the system TR, we obtain an upper triangular system $\mathrm{TR}' = f_1', \ldots, f_h'$ where for $i = 1 \ldots k$, the leading coefficient I_i' of f_i' with respect to y_{k-c+1} is obtained from the leading coefficient I_i of f_i with respect to x_c, the leading variable of f_i.
Moreover, for all i, $i = 1 \ldots k$, the partial derivative of f_i' with respect to the leading variable of f_i' is a multivariate polynomial and not identical to zero. Therefore, there exist infinitely many points, say $P = (\hat{y}_1, \ldots, \hat{y}_k)$ in the affine space $A^k(K)$ such that for all i, $i = 1 \ldots k$, the partial derivative of f_i' with respect to the leading variable of f_i' evaluated at P is not zero; hence, the corresponding Jacobian matrix has full rank. $\square$

REMARK: We have a constructive way to compute such a point as a starting point for the extended Newton's method: (1) Assign arbitrary values to all variables except the leading variable v such that the leading coefficients do not vanish. Note that we might have to do this recursively. (2) Since any univariate polynomial has only finitely many roots, there is a bound for the roots of the polynomial. We assign an arbitrary nonzero value to v and double its value if the polynomial evaluated at the value is not different from zero.
Once we have found a starting point for the extended Newton's method, i.e. a point where the Jacobian matrix has full rank. Under some conditions as in Theorem 4.4.3, all of the subsequent iterates of the method are still defined with the Jacobian matrix evaluated at those points has full rank. That means, all of the subsequent iterates would not be a zero of the initials of the polynomials in TR. Therefore, if the iterates converge, then they will converge to a root of the closure of $\mathcal{V}(\mathrm{CS}) \setminus \mathcal{V}(J)$; and hence a root of PS.

However, in some applications such as CAGD, Visualization, etc., where we would like to find some roots in each irreducible component of the zero set of a system of equations, we need to compute an irreducible characteristic set for each irreducible component of the system. We also need to compute irreducible extended characteristic sets for checking the emptiness of a zero set. Note that it is much more efficient to compute an irreducible characteristic set for each irreducible component of a zero set than to decompose the zero set into the irreducible ones (see Section 2.1).

4.2.2 Main Algorithm

We find a fixed point of the function g with respect to an initial point $x^{(0)}$ by the following algorithm:

Algorithm 4.2.3 (Main)

Inputs: A function $f : \mathbb{C}^k \rightarrow \mathbb{C}^h$, *an initial point* $x^{(0)}$, *a tolerance* ϵ *and a maximum number of iterates* M.

Output: A complex fixed point of the function g.

1. $X := x^{(0)}$; $L := 0$;

2. **If** $h = k$ *and* J *is invertible* **then**
   $g(x) = x - J^{-1} \cdot f(x)$

3. **else**
   if f *is a regular system* **then**
   $g(x) = x - J^{\pm} \cdot f(x)$
   else
   regularize *the system.*

if *"1" is in the regularized system then*
Return(*"the system has an empty zero set"*)

4. $Y := g(X)$;
5. **While** $||Y - X|| \geq \epsilon$ **do**
   (a) *If* $L > M$ *then* **Return**(*"failure"*).
   (b) $X := Y$;
   (c) $Y := g(X)$;
   (d) $L := L + 1$;
6. **Return**(Y).

When the sequence $\{x^{(0)}, x^{(1)}, \ldots, x^{(n)}, \ldots\}$ converges to the fixed point of g, say α in a neighborhood of $x^{(0)}$, we will have $f(\alpha) = 0$. Note that, in general, if we do not regularize the system then we may not have $f(\alpha) = 0$ because from Equation (4.5), if α is a fixed point of the function g then $J^{\pm} f(\alpha) = \vec{0}$. But that condition is not enough to prove that α is a root of the function f; for instance, $\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} = \vec{0}$. Fortunately, this is not the case for an underdetermined system of equations which has a full-rank Jacobian matrix, i.e. $h \leq k$ and $\mathrm{rank} J(x) = h$ (see Corollary 4.2.1).

When $h = k$, in case $J^{\pm}$ and J^{-1} exist, we have $J^{\pm} = J^{-1}$. That means our method is similar to Newton's method when the system has n equations in n variables.

4.3 Some Applications

We have implemented the method using the kernel of the language Maple. All the experiments were carried out on a SGI-IRIS workstation under IRIX 5.2 with 64MB RAM using MIPS-RISC 4000A/100MHz CPU.

4.3.1 Finding the Roots of a System of Transcendental Equations

Given a function $f : \mathbb{C}^2 \to \mathbb{C}$ defined by

$$f(x, y) = x\cos(y) + y\sin(x).$$

The half-inverse Jacobian matrix of f is:

$$J^{\pm} = \begin{pmatrix} \frac{y\cos(x)+\cos(y)}{(y\cos(x)+\cos(y))^2+(\sin(x)-x\sin(y))^2} \\ \\ \frac{\sin(x)-x\sin(y)}{(y\cos(x)+\cos(y))^2+(\sin(x)-x\sin(y))^2} \end{pmatrix}.$$

The iterative function g is:

$$g(x, y) = \begin{pmatrix} x - \frac{(y\cos(x)+\cos(y))\,(x\cos(y)+y\sin(x))}{(y\cos(x)+\cos(y))^2+(\sin(x)-x\sin(y))^2} \\ \\ y - \frac{(x\cos(y)+y\sin(x))\,(\sin(x)-x\sin(y))}{(y\cos(x)+\cos(y))^2+(\sin(x)-x\sin(y))^2} \end{pmatrix}.$$

Table 4.1 shows some roots of the function f which have been obtained by our new method. Note that by using hybrid symbolic–numerical tools, we can get roots with arbitrary precision.

Table 4.1: Example 4.3.1

starting point	timing (sec)	approximate root	accu.
$(0,5)$	0.017	$(0,5)$	0
$(6,-6)$	0.267	$(7.34948123963176133, -5.41311125686218739)$	10^{-18}
$(-10,1)$	0.233	$(-10.0117584262589476, 1.48836671956900328)$	10^{-17}
$(-2+8\,i,$ $-3+10\,i)$	1.800	$(.210^{-42},$ $-2.55629605210303979 + 1.20170464536302136\,i)$	10^{-42}
$(-1+8\,i,$ $1-i)$	1.116	$(-.210^{-35} + .310^{-35}\,i,$ $4.03853685996399011 + .238932104026518823\,i)$	10^{-35}

4.3.2 Finding the Roots of a System of Nonlinear Algebraic Equations

Given a function $f : \mathbb{C}^6 \to \mathbb{C}^3$ defined by

$$f(x_1,\ldots,x_6) = \begin{pmatrix} f_1(x_1,\ldots,x_6) \\ f_2(x_1,\ldots,x_6) \\ f_3(x_1,\ldots,x_6) \end{pmatrix} = \begin{pmatrix} x_1^2 + x_2^2 + x_3^2 - 1, \\ x_4x_1 + x_5x_2 + x_6x_3, \\ (x_4+3x_3)^2 + x_5^2 + (x_6-3x_1)^2 - 1 \end{pmatrix}.$$

Table 4.2 shows some roots of the system of equations f which have been obtained

Table 4.2: Example 4.3.2

starting point	timing (sec)	approximate root	accu.
$(-7,2,8,$ $-6,4,-10)$	19.43	$(-2.336667951, 0.05939562053, 3.242176545,$ $0.8104844811, -0.03433423925, 0.5847529928)$	10^{-8}
$(1,2,-10,$ $4,4,-2)$	20.85	$(0.2251790750, 0.1779114048\,10^{-3}, -3.992859270,$ $-0.9981486227, 0.02303853956, -0.05629001012$	10^{-8}
$(3,8,2,$ $-5,-4,4)$	17.75	$(-2.044477720, 0.9721574287, 2.425304783,$ $0.7312929289, -0.1316558956, 0.6692364289)$	10^{-8}
$(-4+3\,i,$ $-6-4\,i,$ $-1+10\,i,$ $-1-7\,i,$ $9-I,$ $2-7\,i)$	70.16	$(-.28262448798 + 1.230553137837\,i,$ $-2.974233486123 - 1.19345175083\,i,$ $-1.250862534 + 2.559689513217\,i,$ $-2.019668122915 - 10.66954455156\,i,$ $13.08707094054 - 2.72668504254\,i,$ $-2.18788601561 - 10.05636726297\,i)$	10^{-8}
$(-6+8\,i,$ $-10+4\,i,$ $2+i,$ $4-10\,i,$ $-2+4\,i,$ $10)$	68.35	$(1.3890576959 + .3939094985424\,i,$ $-2.652670191716 - 1.566676434031\,i,$ $-1.653457618684 + 2.844366089482\,i,$ $5.417280805 - 9.598523464661\,i,$ $-.2395081590634 + 4.4993702903\,i,$ $8.881353392358 + 1.513629810675\,i)$	10^{-8}

by our new method. Note that by using hybrid symbolic–numerical tools, we can get roots with arbitrary precision.

4.4 Correctness and Convergence of the Algorithm

Given a function $f : \mathbb{C}^k \to \mathbb{C}^h$, and a point $x^{(0)} = [x_1^{(0)}, x_2^{(0)} \dots x_k^{(0)}]^T$, we construct a sequence:

$$x^{(n+1)} = g(x^{(n)}) \tag{4.6}$$

where g is defined by Equation (4.5). Now we consider some conditions in which the sequence will converge:

Theorem 4.4.1 *Let $g(x)$ satisfy $||g(x) - g(y)|| \leq \lambda ||x - y||$ for all vectors x, y such that $||x - x^{(0)}|| \leq \rho$, $||y - x^{(0)}|| \leq \rho$ with the Lipschitz constant, λ, satisfying $0 \leq \lambda < 1$. Let the initial iterate, $x^{(0)}$, satisfy $||g(x^{(0)}) - x^{(0)}|| \leq (1 - \lambda) \cdot \rho$ Then*

(i) all iterates, as in (4.6), satisfy $||x - x^{(0)}|| \leq \rho$;

(ii) the iterates converge to some vector, say α which is a fixed point of g;

(iii) α is the only fixed point of g in $||x - x^{(0)}|| \leq \rho$.

[Note that in general, we may not have $f(\alpha) = 0$].

PROOF. See [Tra94a]. □

Theorem 4.4.1 means that if we are lucky to find a starting point $x^{(0)}$ which satisfies the theorem's conditions, then the iterates converge to a fixed point of g. In other words, the convergence of the iterates in this case depends not only on the function g itself but also on the initial iterate. The problem is that we do not have a constructive way to find such an initial iterate. Hence, we will try to find conditions for the convergence of the iterates which do not heavily depend on the choice of the initial iterate.

In practice, sometime we know in advance that the function g does have a fixed point. With this additional condition, we can find less restrictive conditions on the choice of the initial iterate.

Theorem 4.4.2 *Let $g(x)$ have a fixed point, say α. Let the components $g_i(x)$ have continuous first partial derivatives and satisfy*

$$\left| \frac{\partial g_i(x)}{\partial x_j} \right| \leq \frac{\lambda}{k}, \quad \lambda < 1; \tag{4.7}$$

for all x in

$$||x - \alpha||_\infty \leq \rho \tag{4.8}$$

Then:

(i) For any $x^{(0)}$ satisfying (4.8) all the iterates $x^{(n)}$ of (4.6) also satisfy (4.8).

(ii) For any $x^{(0)}$ satisfying (4.8), the iterates converge to the root α of $x = g(x)$ which is unique in (4.8).

PROOF. For any two points x, y in (4.8) we have by Taylor's theorem: $g_i(x) - g_i(y) = \sum_{j=1}^k \frac{\partial g_i(\xi^{(i)})}{\partial x_j} \cdot (x_j - y_j), \quad i = 1 \dots k$, where $\xi^{(i)}$ is a point on the open line segment joining x and y. Thus $\xi^{(i)}$ is in (4.8), and using (4.7) yields

$$\begin{aligned} |g_i(x) - g_i(y)| &\leq \sum_{j=1}^k \left| \frac{\partial g_i(\xi^{(i)})}{\partial x_j} \right| \cdot |x_j - y_j| \leq ||x - y||_\infty \cdot \sum_{j=1}^k \left| \frac{\partial g_i(\xi^{(i)})}{\partial x_j} \right| \\ &\leq \lambda \cdot ||x - y||_\infty . \end{aligned}$$

Since the inequality holds for each i, we have $||g(x) - g(y)||_\infty \leq \lambda \cdot ||x - y||_\infty$, and thus we have proven that $g(x)$ is Lipschitz continuous in the domain (4.8) with respect to the indicated norm. Now note that for any $x^{(0)}$ in (4.8),

$$||x^{(1)} - \alpha||_\infty = ||g(x^{(0)}) - g(\alpha)||_\infty \leq \lambda \cdot ||x^{(0)} - \alpha||_\infty \leq \lambda \cdot \rho$$

and so $x^{(1)}$ is also in (4.8). By an obvious induction we have then

$$||x^{(n)} - \alpha||_\infty = ||g(x^{(n-1)}) - g(\alpha)||_\infty \leq \lambda \cdot ||x^{(n-1)} - \alpha||_\infty \leq \ldots \leq \lambda^n \cdot \rho. \quad (4.9)$$

and hence all $x^{(n)}$ is also in (4.8). The convergence immediately follows from (4.9) since $\lambda < 1$. The uniqueness follows as in the previous theorem. □

We have the following corollary for other norms.

Corollary 4.4.1 *Theorem 4.4.2 is still true for sum-norm and p-norm.*

Proof. It is obvious for sum-norm. For p-norm, using Hölder's inequality and (4.7) yields

$$\begin{aligned} |g_i(x) - g_i(y)| &\leq \sum_{j=1}^{k} \left| \frac{\partial g_i(\xi^{(i)})}{\partial x_j} \right| \cdot |x_j - y_j| \\ &\leq \sqrt[q]{\sum_{j=1}^{k} \left| \frac{\partial g_i(\xi^{(i)})}{\partial x_j} \right|^q} \cdot \sqrt[p]{\sum_{j=1}^{k} |x_j - y_j|^p} \\ &\leq \frac{\lambda}{\sqrt[p]{k}} \cdot ||x - y||_p. \end{aligned}$$

where $p \geq 1$, $q \geq 1$, $p + q = p \cdot q$. Since the inequality holds for each i, we have

$$||g(x) - g(y)||_p \leq \sqrt[p]{\sum_{i=1}^{k} \left(\frac{\lambda}{\sqrt[p]{k}} \cdot ||x - y||_p \right)^p} \leq \lambda \cdot ||x - y||_p$$

Other details are similar to the proof of the previous theorem. □

If the initial iterate $x^{(0)}$ is sufficiently close to a root α of $f(x) = 0$ where f is regular, then the previous theorems can be used to prove that: The iterates $x^{(n)}$ which are defined in (4.6) converge to the root. However, we do not know from these results if a given initial iterate $x^{(0)}$ is close enough to the unknown root α. In fact, from Theorems 4.4.1 and 4.4.2, we can derive some conditions for the method to converge which depend only on properties of the original function f instead of g; and these conditions may be explicitly checked without knowledge of α.

Theorem 4.4.3 *Let the initial iterate $x^{(0)}$ be such that the Jacobian matrix $J(x^{(0)})$ defined in (4.3) has a half-inverse with norm bounded by*

$$||J^{\pm}(x^{(0)})|| \leq a \quad (4.10)$$

Let the difference of the first two iterates be bounded by

$$||x^{(1)} - x^{(0)}|| = ||J^{\pm}(x^{(0)}) \cdot f(x^{(0)})|| \leq b \quad (4.11)$$

Let the components of $f(x)$ have continuous second derivatives which for all x in $||x - x^{(0)}|| \leq 2b$; $i, j = 1 \ldots k$, satisfy

$$\sum_{j'=1}^{k} \left| \frac{\partial^2 f_i(x)}{\partial x_j \, \partial x_{j'}} \right| \leq \frac{c}{k}, \quad (4.12)$$

Let the Jacobian matrix $J(x^{(n)})$ have a norm bound on its half-inverse (which always exists)

$$||J^{\pm}(x^{(n)})|| \leq ||(I - A_n)^{-1}||.||J^{\pm}(x^{(n-1)})|| \tag{4.13}$$

for all $n = 1, 2, \ldots$; where $A_n = J^{\pm}(x^{(n-1)})\ (J(x^{(n-1)}) - J(x^{(n)}))$, and therefore $J(x^{(n)}) = J(x^{(n-1)})\ (I - A_n)$.
If the constants a, b and c are such that

$$a \cdot b \cdot c \leq 1/2, \tag{4.14}$$

then we have the following:

(i) The iterates are uniquely defined and lie in the "2b-sphere" around $x^{(0)}$:

$$||x^{(n)} - x^{(0)}|| \leq 2b.$$

(ii) The iterates converge to some vectors, say $\lim_{n\to\infty} x^{(n)} = \alpha$, for which $f(\alpha) = 0$, and

$$||x^{(n)} - \alpha|| \leq \frac{2b}{2^n}. \tag{4.15}$$

[All vector norms in the statement and proof of this theorem are max-norms, i.e., $||x|| = \max_i |x_i|$; and matrix norms are the corresponding induced natural norm, i.e., $||A|| = \max_i \left(\sum_{j=1}^{h} |a_{ij}|\right)$.]

PROOF. We prove the theorem by induction. For convenience, we use the notation $J_n \equiv J_f(x^{(n)})$ and show for all $n = 0, 1, 2, \ldots$ that

$$||x^{(n+1)} - x^{(n)}|| \leq \tfrac{b}{2^n}, \tag{4.16}$$

$$||x^{(n+1)} - x^{(0)}|| \leq 2b, \tag{4.17}$$

$$||A_{n+1}|| = ||J_n^{+}.(J_n - J_{n+1})|| \leq \tfrac{1}{2}, \tag{4.18}$$

$$||J_{n+1}^{+}|| \leq \tfrac{||J_n^{+}||}{1-||A_{n+1}||} \leq 2^{n+1}a. \tag{4.19}$$

From the hypotheses, it trivially follows that (4.16, 4.17) are satisfied for $n = 0$. Now when (4.17) is established up to and including any value n then $x^{(n+1)}$ and $x^{(n)}$ are in the $2b$-sphere around $x^{(0)}$ in which we are assured that the second derivatives of the $f_i(x)$ are continuous. Then we can apply Taylor's theorem to the components of J_{n+1} to obtain

$$\frac{\partial f_i(x^{(n+1)})}{\partial x_j} = \frac{\partial f_i(x^{(n)})}{\partial x_j} + \sum_{k=1}^{n} (x_k^{(n+1)} - x_k^{(n)}) \frac{\partial^2 f_i[x^{(n)} + \theta_i(x^{(n+1)} - x^{(n)})]}{\partial x_j \partial x_k},$$

where $0 < \theta_i < 1$.

Since $x^{(n+1)}$ and $x^{(n)}$ are in the $2b$-sphere around $x^{(0)}$, so is the point $x^{(n)} + \theta_i(x^{(n+1)} - x^{(n)})$, and (4.12) can be applied. This gives from the above

$$||J_{n+1} - J_n|| \leq c \cdot ||x^{(n+1)} - x^{(n)}|| \tag{4.20}$$

However, at the present stage in the proof this is valid only for $n = 0$. But then using this and (4.10, 4.11, 4.14) in (4.18) with $n = 0$ yields

$$||A_1|| \leq ||J_0^{+}|| \cdot ||J_1 - J_0|| \leq abc \leq 1/2$$

That means, (4.16, 4.17, 4.18) have been established for $n = 0$.

We have that for any n, if $||A_{n+1}|| < 1$ then A_{n+1} is convergent, $(I - A_{n+1})$ is nonsingular and $||(I - A_{n+1})^{-1}|| \leq \frac{1}{1-||A_{n+1}||}$. In addition, if J_n is a matrix of rank h (i.e. $J^{\pm}(x^{(n)})$ exists.) then J_{n+1} is also a matrix of rank h (i.e. $J^{\pm}(x^{(n+1)})$ exists.). Moreover, we have

$$||J_{n+1}^{+}|| \leq \frac{||J_n^{+}||}{1 - ||A_{n+1}||} \tag{4.21}$$

Since (4.18) is valid for $n = 0$ we get $||J_1^{+}|| \leq 2a$. That means, (4.19) has been verified for $n = 0$.

Let us now make the inductive assumption that (4.16, 4.17, 4.18, 4.19) are valid for all $n \leq k - 1$ and proceed to show that they are also valid for $n = k$.

Since $x^{(k+1)} = x^{(k)} - J_k^{+} f(x^{(k)})$ we have $f(x^{(k)}) = J_k \cdot (x^{(k)} - x^{(k+1)})$ and

$$||x^{(k+1)} - x^{(k)}|| = ||J_k^{+} \cdot f(x^{(k)})|| \leq ||J_k^{+}|| \cdot ||f(x^{(k)})|| \tag{4.22}$$

However, since (4.17) is valid for $n = k-1$, the point $x^{(k)}$ is in the $2b$-sphere around $x^{(0)}$. Then by Taylor's theorem, with remainder term R, $f(x^{(k)}) = f(x^{(k-1)}) + J_{k-1} \cdot [x^{(k)} - x^{(k-1)}] + R(x^{(k)}, x^{(k-1)}) = R(x^{(k)}, x^{(k-1)})$. Using (4.12), we can bound the above remainder term to yield

$$\begin{aligned} ||f(x^{(k)})|| &= \max_i \Bigg| \sum_{j=1}^{n} \sum_{l=1}^{n} \frac{(x_j^{(k)} - x_j^{(k-1)})(x_l^{(k)} - x_l^{(k-1)})}{2!} \cdot \\ &\qquad \frac{\partial^2 f_i}{\partial x_j \partial x_l}[x^{(k-1)} + \phi_i(x^{(k)} - x^{(k-1)})] \Bigg| \\ &\leq \frac{c}{2} ||x^{(k)} - x^{(k-1)}||^2. \end{aligned} \tag{4.23}$$

Again, we have used the fact that $x^{(k-1)} + \phi_i(x^{(k)} - x^{(k-1)})$ is in the $2b$-sphere around $x^{(0)}$ since $x^{(k)}$ and $x^{(k-1)}$ are in it. Now using (4.23) in (4.22) we get

$$||x^{(k+1)} - x^{(k)}|| \leq \frac{c}{2} ||J_k^{+}|| \cdot ||x^{(k)} - x^{(k-1)}||^2 \leq \frac{c}{2}(2^k a) \left(\frac{b}{2^{k-1}}\right)^2 = \frac{ab^2 c}{2^{k-1}} \leq \frac{b}{2^k} \tag{4.24}$$

Thus (4.16) is established. Then since

$$||x^{(k+1)} - x^{(0)}|| = \left\| \sum_{l=0}^{k} (x^{(l+1)} - x^{(l)}) \right\| \leq \sum_{l=0}^{k} ||x^{(l+1)} - x^{(l)}|| \leq b \sum_{l=0}^{k} \frac{1}{2^l} \leq 2b, \tag{4.25}$$

we have also established (4.17). But then $x^{(k+1)}$ is in the $2b$-sphere around $x^{(0)}$. This gives

$$\begin{aligned} ||A_{k+1}|| &= ||J_k^{+} \cdot (J_k - J_{k+1})|| \leq ||J_k^{+}|| \cdot ||J_k - J_{k+1}|| \\ &\leq c||J_k^{+}|| \cdot ||x^{(k+1)} - x^{(k)}|| \leq abc \leq \frac{1}{2}. \end{aligned} \tag{4.26}$$

Then using (4.21) with $n = k$ yields (4.19), and the inductive proof of (4.16, 4.17, 4.18, 4.19) is complete. Part *(i)* of the theorem follows directly from the above facts. The convergence of the $x^{(n)}$ follows from (4.16) since they form a Cauchy sequence:

$$\begin{aligned} ||x^{(n+m)} - x^{(n)}|| &= \left\| \sum_{l=n}^{n+m-1} (x^{(l+1)} - x^{(l)}) \right\| \leq \sum_{l=n}^{n+m-1} ||x^{(l+1)} - x^{(l)}|| \\ &\leq b \sum_{l=n}^{n+m-1} \frac{1}{2^l} \leq \frac{b}{2^{n-1}}. \end{aligned} \tag{4.27}$$

Calling the limit vector α, we use (4.16, 4.23) and the continuity of $f(x)$ to deduce that $||f(x^{(k)})|| \leq \frac{2b^2 c}{4^k}$, and $\lim_{k \to \infty} f(x^{(k)}) = f(\alpha) = 0$. Letting $m \to \infty$, (4.27) yields $||\alpha - x^{(n)}|| \leq \frac{2b}{2^n}$. $\square$

Lemma 4.4.1 *Under the same conditions of Theorem 4.4.2, the convergence of the extended method is quadratic.*

PROOF. Near a root, say α, Equation (4.7) can be satisfied for some $\rho > 0$. If, in addition, $\frac{\partial g_i(\alpha)}{\partial x_j} = 0$, $i, j : 1 \ldots n$, then the second derivatives $\frac{\partial^2 g_i(x)}{\partial x_j \, \partial x_{j'}}$ all exist in a neighborhood of the root, and hence by Taylor's theorem

$$g_i(x) - g_i(\alpha) = \frac{1}{2} \sum_{j=1}^{k} \sum_{j'=1}^{k} \frac{\partial^2 g_i(\xi^i)}{\partial x_j \, \partial x_{j'}} (x_j - \alpha_j)(x_{j'} - \alpha_{j'}).$$

Now in the iteration, we find

$$\|x^{(n)} - \alpha\|_\infty \leq M \|x^{(n-1)} - \alpha\|_\infty^2, \tag{4.28}$$

where M is chosen such that

$$\max_{i,j,j'} \left| \frac{\partial^2 g_i(x)}{\partial x_j \, \partial x_{j'}} \right| \leq \frac{2M}{k^2}.$$

Hence, the convergence of the method is quadratic. $\square$

Part II

Applications in CAGD and Visualization

Chapter 5

Surface-to-Surface Intersections

We present a hybrid symbolic-numerical algorithm for approximation and representation of surface-to-surface intersections – a fundamental and difficult problem in Computer Aided Geometric Design (CAGD) and solid modeling. Reliability and efficiency of intersection algorithms are two basic prerequisites for their effective use in any practical system. Typically, a numerical algorithm is efficient, but it is not fully robust and may fail in certain cases. On the other hand, algorithms based on exact arithmetic are fully robust and accurate, but are normally slow and require a lot of memory space. Perhaps the goals of efficiency and reliability cannot be met simultaneously without some compromises.

In this chapter, we negotiate those compromises judiciously. The key step of the algorithm is based upon a new technique of the author, namely the "Extended Newton Method" for determining the roots of an arbitrary system of equations iteratively, where the equations can be nonlinear algebraic or transcendental. The number of equations of the system can be greater than, less than or equal to the number of variables (see Chapter 4 for more details). Our method for surface-to-surface intersections overcomes the main difficulties of numerical tracing, namely finding a sample point for every real branch and tracing through singularities by using techniques for real root isolation of polynomials with real algebraic number coefficients and the continuation method for solving polynomial systems. On the other hand, the method does not require the expensive computation of birational mappings, projections, reconstructions, etc. as in a pure symbolic method. Furthermore it utilizes floating-point arithmetic whenever possible. Experiments from surface-to-surface intersection examples found in the literature show the considerable speed up of our method in comparison with symbolic methods. Meanwhile, the method still works well near the singularities which is is much better in comparison with numerical methods.

5.1 Introduction

In this chapter, we describe a hybrid symbolic-numerical algorithm for approximation and representation of surface-to-surface intersections – a fundamental and difficult problem in Computer Aided Geometric Design (CAGD) and solid modeling. Given two trivariate polynomials with rational number coefficients, the algorithm produces a space graph which is an approximation and homeomorphic to the real algebraic set of the intersection (in fact, a space curve) in Euclidean space. Due to its importance, there have been persistent efforts in devising algorithms for this problem (see [BFJP87, BHLH88, Hof89, KPP90, W+91], and [Pat92, Pat93]). The main issue in the intersection problem is the efficient discovery and description of all features of the solution with high precision commensurate with the tasks required by the underlying geometric modeler. Reliability and efficiency of intersection algorithms are two basic prerequisites for their effective use in any geometric modeling system. It is closely associated with the way the algorithm handles such features as near singular cases, small loops, etc.

Roughly speaking, surface intersection methods can be classified in two main categories: symbolic approach and numerical approach.

Symbolic methods (as in [Win93, Wal93]) are based upon the primitive element theorem of Algebraic Geometry: every irreducible d-dimensional algebraic set in n-dimensional space is, after a suitable linear transformation of coordinates, birationally projectable onto an irreducible d-dimensional algebraic set in $d+1$-dimensional space, which can be specified by a single polynomial in $d+1$ variables together with the birational mappings. The general structure of algorithms using symbolic approach is:

(1) Decompose the algebraic set of the intersection into the union of irreducible ones by using, for instance, Ritt–Wu's method (see [Wan92]).

(2) For each irreducible algebraic set, project it onto a hypersurface getting a plane curve and a set of birational mappings.

(3) Analyze the topology of the plane real algebraic curves.

(4) Project the plane real algebraic curves back into Euclidean space using their birational mapping.

The method is reliable. It can handle the features near singular cases, small loops, or even partial surface overlap. However, the algorithm is complicated, inefficient and requires a lot of memory space, especially when we have to deal with points with real algebraic number coordinates.

On the other hand, we can trace surface-to-surface intersections numerically (see [BHLH88, BK90, PP90, KP91b, KPPW92]). The method involves generating point sequences of each separated branch of the algebraic set by stepping from a given point on the required branch in a direction prescribed by the curve's local differential geometry. However such methods are by themselves incomplete in that they require starting points for every branch of the solution. Finding the starting points is a difficult task. Moreover, the method is not reliable and has great difficulties when tracing through singularities. The reason stems from this fact: a numerical tracing scheme technically requires formulating and solving a linear system of equations. Such a system is formulated both for determining an approximation to the curve at a given points, as well as when refining an estimate of the location of a point on the curve with Newton iterations. At a curve singularity, the linear system is singular, and nearby the condition number of the system is so large that roundoff errors destroy all accuracy of the estimate.

Our method is a compromise between those two methods that uses exact computation or numerical computation when needed. The key step of the algorithm is based upon a new technique of the author, namely the "Extended Newton Method" for determining the roots of an arbitrary system of equations iteratively, where the equations can be nonlinear algebraic or transcendental, the number of equations of the system can be greater than, less than or equal to the number of variables (see [Tra94b] and Chapter 4 for more details).. Our algorithm for surface-to-surface intersections overcomes the main difficulties of numerical tracing, namely finding a sample point for every real branch and tracing through singularities by using techniques for real root isolation of polynomials with real algebraic number coefficients and the continuation method for solving polynomial systems. On the other hand, the method does not require the expensive computation of birational mappings, projections, reconstructions, etc. as in a pure symbolic method. Further it utilizes floating-point arithmetic whenever possible. Experiments from surface-to-surface intersection examples found in the literature show the considerable speed up of our method in comparison with symbolic methods. Meanwhile, the method still works well near the singularities which is is much better in comparison with numerical methods.

Other methods for tackling the problem which we should mention here are the subdivision method (see [Tau94b]) and Cylindrical Algebraic Decomposition(see [ACM88]).

5.2 Main Algorithm

In this section, we elaborate the main algorithm.

Algorithm 5.2.1 (Main)

Inputs: a one-dimensional algebraic set of the intersection of two surfaces defined implicitly by two trivariate polynomials $p_1(x, y, z)$, $p_2(x, y, z) \in \mathbb{Q}[x, y, z]$. *A bounded box* $\mathcal{S}$ *in which we want to represent the intersection* $\{(x, y, z) \in \mathbb{R}^3 : x_1 \leq x \leq x_2,\ y_1 \leq y \leq y_2,\ z_1 \leq z \leq z_2\}$. *A tolerance* ϵ *for precision of the representation.*

Output: A space graph of the algebraic set $A = \{(x, y, z) \in \mathbb{R}^3 : p_1(x, y, z) = p_2(x, y, z) = 0\}$.

1. *Compute all extremal points (including singularities) of the algebraic set* A *in some direction, say* x, *by using Dixon's resultant method to solve the system of 3 equations in 3 variables* $\{p_1(x, y, z) = 0,\ p_2(x, y, z) = 0,\ \det(J_x) = 0\}$ *where* $J_x = \left(\frac{\partial p_i}{\partial \xi_j}\right)$, $i = 1..2$, $j = 1..2$; $\xi_1 = y$, $\xi_2 = z$ *and* $\det(J_x) \not\equiv 0$.

2. *For each interval* $\mathcal{I}_i$ *on the* x *axis between two extremal points,*

   (a) *Compute the resultant* $A_i(y) = \mathrm{res}_z(p_1(x_i, y, z),\ p_2(x_i, y, z))$ *where* x_i *is the rational middle point of the interval.*

   (b) *Compute the greatest square free divisor,* $\bar{A}_i(y)$, *of* $A_i(y)$.

   (c) *Compute* $\mathrm{Lc} := \mathrm{gcd}(\mathrm{gcd}(\mathrm{lcoeff}_z(p_1(x_i, y, z)), \mathrm{lco}$-$\mathrm{eff}_z(p_2(x_i, y, z))), \bar{A}_i(y))$.

   (d) *Isolate all of the real roots of* $\bar{A}_i(y)$ *by using algorithm based on Descartes' lemma for univariate polynomials with integral coefficients getting a list of intervals* $\mathrm{Ical\_l_i}$

   (e) *Remove intervals in* $\mathrm{Ical\_l_i}$ *which is also an interval for a real root of* Lc.

3. *For all* i, *for each interval* $\mathrm{Ical_{i,j}}$ *in* $\mathrm{Ical\_l_i}$, $\mathrm{Ical_{i,j}}$ *and* $\bar{A}_i(y))$ *define a real algebraic root of* $\bar{A}_i(y))$, *say* α.

   (a) *Compute* $q = \mathrm{gcd}(p_1(x_i, \alpha, z),\ p_2(x_i, \alpha, z))$.

   (b) *Isolate all of the real roots of* $q(x_i, \alpha, z)$ *by using Algorithm 2.3.2 for univariate polynomials with real algebraic coefficients up to the precision* ϵ, *getting a list of intervals* $\mathrm{Ivl\_l_{i,j}}$.

4. *From each initial point defined by a member of the list of intervals* $\mathrm{Ivl\_l_{i,j}}$, *trace the simple path of the intersection, which is defined by a system of two equations in three variables, by using the "Extended Newton Method" to the nearby two extremal points on two directions.*

5. Represent the space curve.

Remark on the input specification

When the intersection is not simple, the ranges for variables can automatically be defined from the extremal points, which are finite. The tolerance ϵ can also be defined from the ranges for variables and the size of the pixel which is fixed for each screen.

Remark on Step 1

We can have an intersection without any extremal point or singularity, namely the simple space curve. In [Tra95b], we use the continuation method in this step. The disadvantage of the continuation method is that we have to compute all of the complex roots. The same idea of this step has been used in [KP91a]. We use a numerical method with appropriate length of mantissa in this step because we do not need to isolate points which lie in the same pixel of the screen when we plot them. Meanwhile, an exact method will spend a lot of time and requires a huge amount of memory space for isolation of very near points which will be plotted exactly the same on the screen.

Remark on Step 2

We have to take into the account the cases when the $\mathrm{l}coeff_z($ $p_1(x_i, y, z))$ and $\mathrm{l}coeff_z(p_2(x_i, y, z))$ which are polynomial in y have a common real root. In fact, we do not need to compute the primitive part of $p_1(x_i,\ y, z)$ and $p_2(x_i, y, z)$ because they will not have a common factor which is a univariate polynomial in y (otherwise, the intersection will have dimension greater then 1).

Remark on Step 3

The step, in fact, computes the starting point for all paths of the intersection between two extremal points. Therefore, we have to isolate them exactly. Otherwise we could loss some paths of the intersection.

Remark on Step 4

Every path between two extremal points is simple. That means that we do not have to trace through singularities.

5.3 Correctness and Convergence of the Main Algorithm

In this section, we will prove some theorems which are concerned with the correctness and convergence of the main algorithm.

In Step 3, we use the most efficient algorithm for isolating the roots of a polynomial namely the method based on Descartes' lemma. But this method requires that the polynomial has to be square free. Meanwhile the greatest square free divisor computation is very expensive for polynomials with real algebraic coefficients. Fortunately, we can avoid that expensive computation. In fact, the correctness of the step is guaranteed by the following theorem.

Theorem 5.3.1 *For all $\mathcal{I}_i$, for all α, $q(x_i, \alpha, z)$ is square free in $\mathbb{Z}[\alpha][z]$.*

Proof: Otherwise the intersection would have an extremal point at $x = x_i$.□

Meanwhile, the correctness and convergence of Step 4 come from the following theorems.

5.4 Use of Floating-point Arithmetic

The most time consuming part of the real root isolation algorithm presented in Section 2.3 is that of sign computations of real algebraic numbers (see [Joh91, Hon93]). In fact, at least theoretically, the time for the sign computations dominates the cost of the entire algorithm. In this section, we briefly describe how floating-point numbers can be used in order to speed up these computations. First, to reduce the number of sign computations, we

(1) Approximate α by floating-point number (by repeatedly bisecting its isolating interval).

(2) Numerically evaluate $P(x, y)$ at $x = \alpha$.

(3) Numerically guess the real roots of $P(\alpha, y)$ (by using Laguerre's method to approximate all the complex roots and filtering out nonreal-roots).

(4) Form rational intervals for the real roots we guessed.

(5) Verify/correct them by the exact method. □

We further reduce the time by using hardware floating-point arithmetic. Let $P \in \mathbb{Q}(\alpha)$ where α is a real root of a minimal polynomial A in an interval I. Let us assume that P is presented as an element of $\mathbb{Q}[x]$. Then we do the following:

(1) Refine I

(2) Evaluate P on I using interval arithmetic.

(3) Test the sign of the resulting interval

(4) If the sign cannot be determined, then use exact method.

For details on numeric and interval methods, see [Act70, Moo79].

5.5 Experiments

We have implemented the algorithm using the kernel of the Maple language. All the experiments were carried out on a SGI-IRIS workstation under IRIX 5.2 with 64MB RAM using MIPS-RISC 4000A/100MHz CPU. Figure 5.1 shows the graphs produced by the program for the following examples taken from literature.

1. $x - 1$, $y - 1$
2. $x^2 + y^2 + z^2 - 5$, $xyz - 1$
3. $z - 2x^4 - y^4$, $z - 3x^2y + y^2 - x^3$
4. $z - 2x^4 - y^4$, $z^2 - 3x^2y - y^3$
5. $z^2 + 3x^4 - y^4$, $z + x^3 - 3x^2y + y^2 - 2y^3$
6. $z^3 + 7xy^2 - x^3 - x^2z + 6$, $z^2 - 8x^3 + xy^2 + xyz$

Figure 5.2 reports the time comparison between the new hybrid method and a pure symbolic method which is implemented in CASA, a system for constructive algebraic geometry developed at RISC-Linz (see [Win93, Wal93]).

Figure 5.1: Surface-to-Surface Intersections

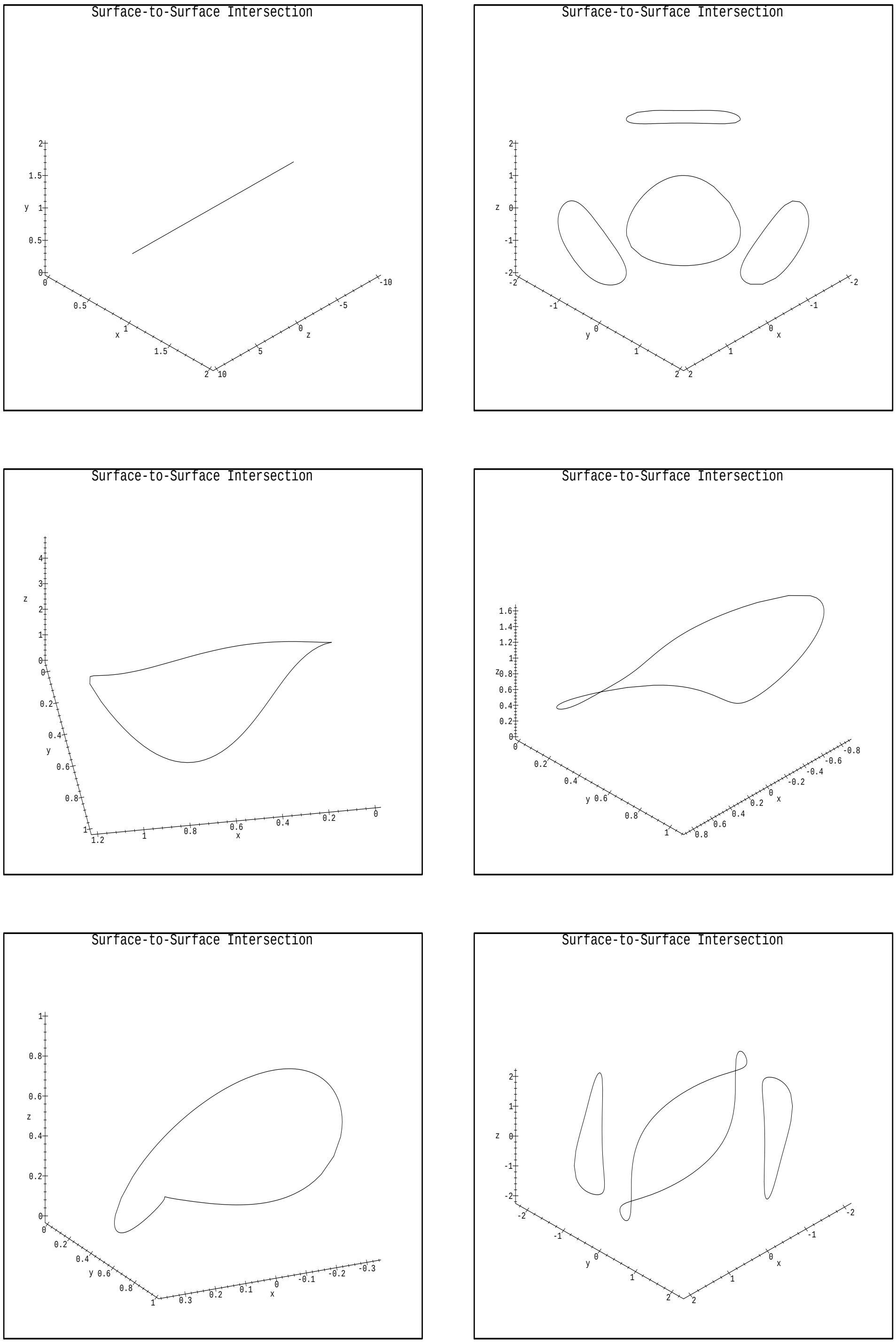


Figure 5.2: Time Comparison

	Exact Method (second)	Hybrid Method (second)
Example 1	3.483	0.366
Example 2	169.050	27.083
Example 3	> 10000.000	15.850
Example 4	> 10000.000	33.400
Example 5	> 50000.000	144.217
Example 6	> 100000.000	1760.367

Chapter 6

Visualization of Curves in Higher Dimensions

We present a hybrid symbolic-numerical method for tracing along a one-dimensional algebraic set in higher dimensions. The key step of the algorithm is based upon a new result of the author's on pseudoirreducible decomposition (see Section 2.2 for more details) and an iterative method of the author's for finding the roots of an arbitrary system of equations, namely the "Extended Newton Method" (see Chapter 4 for more details). In that, given a one-dimensional algebraic set in higher dimensions, which is reducible or not, in form of n equations in m variables, the algorithm will compute, without the expensive computation of irreducible decomposition, a starting point for each pseudoirreducible component of the algebraic set. By pseudoirreducible set of a one-dimensional irreducible algebraic set $\mathcal{V}$, we mean an affine set which is coincided with $\mathcal{V}$ except at finitely many points (in general case, it is coincided with $\mathcal{V}$ except at a measure-zero set of points). Our algorithm for one-dimensional algebraic set in higher dimensions overcomes the main difficulties of numerical tracing, namely finding a sample point for every real branch. The method does not require the expensive computation of irreducible decomposition. Moreover, it will prevent duplicate tracing and additional singularities as we may have when we trace a reducible space curve. The method also overcomes the difficulties of roundoff errors near the singularities and component jumping. Further it utilizes floating-point and parallel arithmetic whenever possible. Experiments from examples found in the literature shows the considerable speed up of the method.

6.1 Introduction

In this chapter, we present a hybrid symbolic-numerical method for tracing along a one-dimensional algebraic set in higher dimensions. Approximation and representation of one-dimensional algebraic sets in higher dimensions is a fundamental and difficult problem in Computer Aided Geometric Design (CAGD) and solid modeling. For example, the intersection of two parametric surfaces can be seen as a space curve in 4-dimensional space and the intersection of an implicit surface with an r-offset surface of another implicit surface can be seen as a space curve in 6-dimensional space. Due to its importance, there have been persistent efforts in devising algorithms for this problem (see [BFJP87], [Hof89], [W$^+$91], [Pat92], [Pat93], [KM95]). The main issue in the problem is the efficient discovery and description of all features of the solution with high precision commensurate with the tasks required from the underlying geometric modeler. Reliability and efficiency of intersection algorithms are two basic prerequisites for their effective use in any geometric modeling system.

It is closely associated with the way the algorithm handles such features as near singular cases, small loops, etc.

In higher dimensional spaces, symbolic methods by decomposing the algebraic set of the intersection into the union of irreducible ones and birational projecting each of them onto a hypersurface getting a plane curve and a set of birational mappings are extremely inefficient and requires a lot of memory space. Especially, when we have to deal with points with real algebraic number coordinates. Beside of that, the very high degree of the projected plane algebraic curve, which is, in general, the product of the surfaces' degree, leads to difficulties when analyzing the topology of the plane algebraic curves.

On the other hand, we can trace one-dimensional algebraic sets numerically (see [BHLH88], [BK90], [PP90], [KP91b]). The general structure of algorithms using this approach is:

(1) Find an initial point on the curve.

(2) Obtain an estimate of the next curve point by constructing a local approximation of the curve and by stepping along the approximation a specific distance.

(3) Refine the new point estimate to a curve point.

The method involves generating a sequence of points for each separated branch of the algebraic set by stepping from a given point on the required branch in a direction prescribed by the curve's local differential geometry. The advantages of this approach are that we can directly trace a curve in higher dimensions and the algebraic degrees involved are often low, which will increase the numerical accuracy of the method. However such methods are by themselves incomplete in that they require starting points for every branch of the solution. Finding the starting points in real affine space is a difficult task. In most cases, initial starting points for the trace can only be found by the intersections of the curve to be traced with the boundary of a given domain. Finding the starting points in higher dimensions is rather harder since the complexity grows exponentially with the dimension. Beside of that, in case of a reducible one-dimensional algebraic set, several branches of the curve will be duplicately traced and additional singularities might be presented on the curve. Moreover, the method is not reliable and has great difficulties when tracing through singularities. The reason stems from this fact: a numerical tracing scheme technically requires formulating and solving a linear system of equations. Such a system is formulated both for determining an approximation to the curve at a given points, as well as when refining an estimate of the location of a point on the curve with Newton iterations. At a curve singularity, the linear system is singular, and nearby the condition number of the system is so large that roundoff errors destroy all accuracy of the estimate.

In Chapter 5 and in [Tra95a], the author presented a hybrid method for tracing the intersections of two implicit surfaces in 3-dimensions. That method overcomes the main difficulties of numerical tracing, namely finding a sample point for every real branch and marching through singularities by using techniques for real root isolation of polynomials with real algebraic number coefficients and the continuation method for solving polynomial systems. However, that method would lose its advantages in higher dimensions because we have to use much more symbolic computations in these cases.

Our method is a hybrid method that uses exact computation or numerical computation when needed. The key step of the algorithm is based upon a combination of dimension theory, characteristic sets method and an iterative method of the author's for finding the roots of an arbitrary system of equations, namely the "Extended Newton Method" (see Chapter 4, [Tra94b], [Tra95a] and [Tra96a] for more details). In that, given a one-dimensional algebraic set in higher dimensions, which is reducible or not, in form of n equations in m variables, the algorithm will compute, without the expensive computation of irreducible decomposition, a starting point for each pseudoirreducible components of the algebraic set. By pseudoirreducible set of a one-dimensional irreducible algebraic set $\mathcal{V}$, we mean an affine set which is coincided with $\mathcal{V}$ except at finitely many points (in general case, it is coincided with $\mathcal{V}$ except at a measure-zero set of points). Our algorithm for one-dimensional algebraic set in higher dimensions overcomes the main difficulties of numerical tracing, namely finding a sample point for every real branch. The method does not require the expensive computation of irreducible decomposition. Moreover, it will prevent duplicate tracing and additional singularities as we may have when we trace a reducible space curve. The method also overcomes the difficulties of roundoff errors near the singularities and component jumping. Further it utilizes floating-point and parallel arithmetic whenever possible. Experiments from examples found in the literature shows the considerable speed up of the method.

The structure of this chapter is as follows. In Section 6.2, we elaborate the main algorithm. In Section 6.3, we show how to achieve further efficiency by the use of floating point and parallel arithmetic. In Section 6.4, we report some experimental results.

6.2 Main Algorithm

In this section, we elaborate the main algorithm. Since the algebraic computations implicitly require projective spaces over an algebraically closed ground field, in this section, sometime we need the ground field to be $\mathbb{C}$, the field of complex numbers. More precisely, we will work over a computable subfield of complex numbers. Of course, we always start with rational coefficient polynomials and try to evaluate the real part of the one-dimensional algebraic sets.

It is known in algebraic geometry that given an algebraic set $\mathcal{V} \subset A^n(\mathbb{K})$, there exists a unique decomposition of $\mathcal{V}$ into irreducible algebraic sets $V_1, V_2, \ldots V_m$ such that

$$\mathcal{V} = V_1 \cup V_2 \ldots \cup V_m,\ \forall i \neq j : V_i \not\subseteq V_j.$$

A usual algorithm to compute the irreducible decomposition of an algebraic set $\mathcal{V}(PS)$ corresponding to a system PS of polynomials in $\mathbb{K}[x_1, x_2 \ldots, x_n]$ is as follows.

1. Compute a sequence of irreducible characteristic sets $CS_1, CS_2, \ldots, CS_m$ (also called characteristic set series). We can assume that the sequence is nonredundant, i.e., $\forall i \neq j,\ \exists f \in C_i : \mathrm{prem}(f, CS_j) \neq 0$
2. For each irreducible characteristic set CS_i, compute Gröbner basis GB_i of the ideal $\langle f_1, \ldots, f_r, y \cdot I_1 \cdots I_r - 1\rangle$ with respect to the lexicalgraphics ordering.
3. Compute $\psi_i = GB_i \cap \mathbb{K}[x_1, x_2 \ldots, x_n]$. Then $\bigcup_{i=1}^m \mathcal{V}(\psi_i)$ is the irreducible decomposition of $\mathcal{V}(PS)$.

Experiments show that the most heavy computation step of the decomposition algorithm is Step 2 where we have several variables.

In fact, we can avoid such heavy computation by using the new concept of pseudoirreducible presented in Section 2.2. In that, given an algebraic set $\mathcal{V}$, a set of affine sets $\{\overline{V_1}, \ldots, \overline{V_m}\}$ is called a pseudoirreducible decomposition of $\mathcal{V}$ if for all $i = 1 \ldots m$, $\overline{V_i}$ is coincided with V_i except at a measure zero set of points (in comparison with V_i), where $V_1, V_2, \ldots V_m$ is the irreducible decomposition of $\mathcal{V}$. Note that, $\overline{V_i}$ need not be an algebraic set. We also have proved the fact that:

Given a system PS of polynomials with a nonredundant sequence of irreducible characteristic sets CS_1, CS_2, ..., CS_m, then

$$(\mathcal{V}(CS_1) \setminus \mathcal{V}(J_1)), \; \ldots, (\mathcal{V}(CS_m) \setminus \mathcal{V}(J_m))$$

is a pseudoirreducible decomposition of $\mathcal{V}(PS)$

The advantages of this approach are that

1. We almost have the irreducible decomposition.
2. The system of equations corresponding to a pseudoirreducible component is very simple which is much easier to be traced or evaluated.

We have the main algorithm as follows.

Algorithm 6.2.1 (Main)

Inputs: a one-dimensional algebraic set in higher dimensions defined implicitly by m polynomials $p_1, \ldots, p_m \in \mathbb{Q}[x_1, \ldots, x_n]$. A tolerance ϵ for precision of the representation.

Output: A space "graph" of the algebraic set $V = \{(x_1, \ldots, x_n) \in \mathbb{R}^n : p_1(x_1, \ldots, x_n) = \ldots = p_m(x_1, \ldots, x_n) = 0\}$.

1. *Compute a pseudoirreducible decomposition of $\mathcal{V}(PS)$ by using Theorem 2.2.7.*
2. *For each pseudoirreducible component of the algebraic set, find a starting point by using the method in Section 4.2 with some of the variables can take complex values.*
3. *From a point on the curve, an approximate value of the next point is obtained by taking a small step size in a direction determined by the local geometry of the curve. Refine the approximate value to a curve point by using the method in Section 4.2.*
4. *Represent the space curve.*

Remark on the algorithm

One-dimensional algebraic set may consist of open components and loops. In that, finding a starting point for each real loop of the algebraic set is significant hard. However, with the use of a simple algebraic property; more precisely, by extending the ground field, we can overcome the difficulty. If we relax the restriction on the real values of the variables so that some of the variables can take complex values, the curve is now defined as continuous set consisting of real and complex components.

Lemma 6.2.1 *If the one-dimensional algebraic set consists of a real closed component, then two arbitrary complex conjugate paths meet at one of the real points (corresponding to a turning point) on the loop.*

Proof: All of the coefficients of the defining polynomials of the algebraic set are real; hence, the complex roots must occur in conjugate pairs. □

If we want to trace the curve within a bounded box $\mathcal{S} = [U_1^1, U_1^2] \times \cdots \times [U_n^1, U_n^2]$ then we need to compute all of the real and complex intersections of the curve with the boundary of the bounded box, which can be done by, for example, continuation method.

To prevent component jumping, redundant tracing and to handle the roundoff errors near the singularities, we need to recognize the singularities of the curve while tracing. In fact, we can do so by using the fraction free or integer preserving

Gaussian elimination method (see [Bar68]) to eliminate the variables of the Jacobian matrix. Note that by using the fraction free elimination method, the determinant of the Jacobian matrix will come for free. Once the singularities are located, we can prevent redundant tracing by the use of Puiseux expansion. By using techniques in Section 6.3 to evaluate the pseudoinverse Jacobian matrix, we are able to handle the roundoff errors near the singularities. The correctness and convergence of the main algorithm come from Theorem 2.2.7.

6.3 Use of Floating-point and Parallel Arithmetic

In this section, we briefly describe how floating-point numbers and parallel arithmetic can be used in order to speed up the computations of the main algorithm. We also use them to handle the situation near the singularities.

Given an $n \times n$ nonsingular matrix A, we consider the characteristic polynomial ϕ of A, defined by

$$\phi(\lambda) = \det(\lambda I - A) = \prod_{i=1}^{n} (\lambda - \lambda_i) \tag{6.1}$$

where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A. Let $c_1, \ldots, c_n$ be the coefficients of the characteristic polynomial, that is

$$\phi(\lambda) = \lambda^n + c_1 \lambda^{n-1} + \cdots + c_{n-1}\lambda + c_n \tag{6.2}$$

By comparing Equations (6.1) and (6.2), we see that $c_n = (-1)^n \prod_{i=1}^{n} \lambda_i$; in particular, A is invertible if and only if $c_n \neq 0$, which we assume. We will compute A^{-1} using Cayley–Hamilton theorem, which states that $\phi(A) = A^n + c_1 A^{n-1} + \cdots + c_{n-1}A + c_n I = 0$. Therefore, A^{-1} is given by

$$A^{-1} = -\frac{1}{c_n}(A^{n-1} + c_1 A^{n-2} \cdots + c_{n-1} I) \tag{6.3}$$

The matrix $A^2, \ldots, A^{n-1}$ can be all generated in time $O(\log^2 n)$ using n^4 processor. We now use a classical method, known as Leverrier method that allows us to compute the coefficients of the characteristic polynomial in terms of $s_1, \ldots, s_n$, where $s_k = \sum_{i=1}^{n} \lambda_i^k$. In that, the coefficients $c_1, \ldots, c_n$ and the coefficients $s_1, \ldots, s_n$ satisfy the following system of equations:

$$\begin{bmatrix} 1 & 0 & \ldots & \ldots & \ldots & 0 \\ s_1 & 2 & 0 & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ s_{k-1} & \ldots & s_1 & k & 0 & \vdots \\ \vdots & \ddots & & \ddots & \ddots & 0 \\ s_{n-1} & \ldots & s_{k-1} & \ldots & s_1 & n \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ c_n \end{bmatrix} = - \begin{bmatrix} s_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ s_n \end{bmatrix} \tag{6.4}$$

Note that Equation (6.4) is a lower triangular system of equations. Hence, we can solve for $c_1, \ldots, c_n$in time $O(\log^2 n)$ using n^3 processor. We can now summarize the algorithm to compute the pseudoinverse matrix:

1. Compute $(J \cdot J^*)^k$ for $k = 2, \ldots, n$.
2. Compute s_k for $k = 1, \ldots, n$.
3. Solve the system (6.4) for $c_1, \ldots, c_n$.
4. Evaluate $J^* \cdot (J \cdot J^*)^{-1}$ using Equation (6.3).

6.4 Experiments

We have implemented the algorithm using CASA, a computer algebra software for constructive algebraic geometry developed at RISC-Linz and based upon the kernel of the Maple language. All the experiments were carried out on a SGI-IRIS workstation under IRIX 5.2 with 64MB RAM using MIPS-RISC 4000A/100MHz CPU.

The following are two examples of one-dimensional algebraic sets which are very tough for other methods. In fact, the computer system would crash if one tries to compute the irreducible decomposition or Gröbner basis of the ideal corresponding to the algebraic sets.

$$\begin{aligned}
PS_1 &= \{(x-u)^2+(y-v)^2-1, v^2-u^3,\\
&\quad 2v(x-u)+3u^2(y-v), (3wu^2-1)(2wv-1)\}\\
PS_2 &= \{t_1^2+t_2^2+t_3^2-1,\\
&\quad u_1t_1+u_2t_2+u_3t_3,\\
&\quad (u_1+3t_3)^2+u_2^2+(u_3-3t_1)^2-1,\\
&\quad (u_1-3t_3)^2+u_2^2+(u_3+3t_1)^2-1,\\
&\quad (u_1-7t_3+t_2)^2+(u_2-t_1)^2+(u_3+7t_1)^2-1,\\
&\quad (u_1-v_1t_3+2t_2)^2+(u_2-2t_1+t_3)^2+\\
&\quad +(u_3-t_2+v_1t_1)^2-1\}
\end{aligned}$$

Meanwhile, using our method, we can trace along those one-dimensional algebraic sets. In the Figure 6.1, we report the time to use for tracing along the algebraic sets with the tolerance of 10^{-15}. The time for approximation and refining of regular points or near singularities are the average of 1000 tests.

Figure 6.1: Time Consuming

	Tracing PS_1 (second)	Tracing PS_2 (second)
Starting point	52.484	82.500
Regular points	~ 9.384	~ 24.400
Near singularities	~ 20.150	~ 26.666

Bibliography

[ACM88] D. S. Arnon, G. E. Collins, and S. McCallum. An adjacency algorithm for cylindrical algebraic decompositions of three-dimentional space. *J. Symbolic Computation*, 5, 1988.

[Act70] Forman S. Acton. *Numerical Methods that work.* Harper and Row, New York, 1970.

[Atk78] K. E. Atkinson. *An Introduction to numericl Analysis.* John Wiley & Sons, New York, 1978.

[Baj90a] C.L. Bajaj. The emergence of algebraic curves and surfaces in geometric design. Technical Report CSD-TR-92-056, Purdue Univ., Dept. of Comp. Sci., West Lafayette, Indiana, USA, 1990.

[Baj90b] C.L. Bajaj. Geometric modeling with algebraic surfaces. Technical report, Purdue Univ., Dept. of Comp. Sci., West Lafayette, Indiana, USA, 1990.

[Bar68] E. H. Bareiss. Sylvester's identity and multistep integer preserving Gaussian elimination. *Mathematics of Computation*, 22:565–578, 1968.

[Bar85] R. E. Barnhill. Surface in computer aided geometric design: A survey with new results. *Computer Aided Geometric Design*, 2:1–17, 1985.

[BCX94] C. Bajaj, J. Chen, and G. Xu. Free form surface design with a-patches. In *Proc. of Graphics Interface 94*, Banff, Alberta, Canada, 1994.

[BFJP87] R.E. Barnhill, G. Farin, M. Jordan, and B.R. Piper. Surface/surface intersection. *Comp. Aided Geom. Design*, 4:3–16, 1987.

[BFK84] W. Böhm, G. Farin, and J. Kahmann. A survey of curve and surface methods in CAGD. *Comp. Aided Geom. Design*, 1:1–60, 1984.

[BHLH88] C. L. Bajaj, C. M. Hoffmann, R. E. Lynch, and J. E. H. Hopcroft. Tracing Surface Intersections. *Computer Aided Geom. Design*, 5: pages 285–307, 1988.

[BIG74] A. Ben-Israel and T. N. E. Greville. *Generalized Inverses: Theory and Applications.* John Wiley & sons, 1974.

[Bje51a] A. Bjerhammar. Applications of calculus of matrices to method of least squares; wth special reference to geodetic calculations. Technical Report 49, Kungl. Tekn. Högsk. Handl., 1951.

[Bje51b] A. Bjerhammar. Rectangular reciprocal matrices with special reference to geodetic calculations. *Bull. Géodésique*, pages 188–220, 1951.

[Bje68] A. Bjerhammar. A generalized matrix algebra. Technical Report 124, Kungl. Tekn. Högsk. Handl., 1968.

[BK90] R. E. Barnhill and S. N. Kersey. A marching method for parametric surface/surface intersections. *Computer Aided Geometric Design*, 7(1–4):257–280, 1990.

[BO71] T. L. Boullion and P. L. Odell. *Generalized Inverse Matrices.* Wiley–Interscience, New York, 1971.

[BR90] R. Benedetti and Jean-Jacques Risler. *Real Algebraic and Semi-Algebraic Sets.* Hermann, Paris, 1990.

[BX94a] C.L. Bajaj and Guoliang Xu. A-splines: Local interpolation and approximations using C^k continuous piecewise real algebraic curves. Technical report, Purdue Univ., Dept. of Comp. Sci., West Lafayette, Indiana, USA, 1994.

[BX94b] C.L. Bajaj and Guoliang Xu. NURBS approximations of real algebraic curves. Technical report, Purdue Univ., Dept. of Comp. Sci., West Lafayette, Indiana, USA, 1994.

[Can90] J. Canny. Generalized characteristic polynomials. *Journal of Symbolic Computation*, pages 241–250, 1990.

[Cay65] A. Cayley. On the theory of elimination. *Cambridge and Dublin Mathematical Journal*, III:210–270, 1865.

[CG90] S. C. Chou and X. S. Gao. Methods for mechanical geometry formula deriving. In *Proceedings ISSAC 1990*, ACM press, 1990.

[Chi90] E. Chionh. *Base points, resultants, and the implicit representation of rational Surfaces.* PhD thesis, Dept. Comp. Sci., Uni. of Waterloo, 1990.

[Cho88] Shang Ching Chou. *Mechanical Geometry Theorem Proving.* D. Reidel, 1988.

[CLO92] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: an Introduction to Computational Algebraic Geometry and Commutative Algebra.* Undergraduate Texts in Mathematics. Springer–Verlag, 1992.

[Dix08] A.-L. Dixon. The eliminant of three quantics in two independent variables. In *Proc. London Mathematical Society*, volume 6, pages 468–478, 1908.

[Dre78] F.J. Drexler. A homotopy method for the calculation of zeros of zero-dimensional polynomial ideals. In A. G. Wacker, editor, *Continuation Methods*, pages 69–93, New York, 1978. Academic Press.

[EHV90] D. Eisenbud, C. Huneke, and W. Vasconcelos. Direct methods for primary decomposition. *Invent. Math.*, 1990.

[Far88] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design - A Practical Guide.* Academic Press, New York, 1988.

[Fre03] I. Fredholm. Sur une classe d'équations fonctionnelles. *Acta Math.*, 27:365–390, 1903.

[Gre60] T. N. E. Greville. Some applications of the pseudo inverse of a matrix. *SIAM rev.*, II:15–22, 1960.

[GTZ88] P. Gianni, B. M. Trager, and G. Zacharias. Gröbner Bases and Primary Decomposition of Polynomial Ideals. *J. Symb. Comput.*, 6: pages 149–167, 1988.

[GW92] X. S. Gao and D. K. Wang. On the automatic derivation of a set of geometric formulae. In *Mathematics Mechanization Research Preprints*, number 8. Academia Sinica, 1992.

[Her26] G. Hermann. Die Frage der endlich vielen Schritte in der Theorie der Polynomideale. *Math. Annalen*, 95:736–788, 1926.

[Her75] I. N. Herstein. *Topics in Algebra.* John Wiley & Sons, second edition edition, 1975.

[Hil12] D. Hilbert. *Grundzüge einer allgemeinen Theorie der linearen Integralgleichungen.* B. G. Teubner, Leipzig and Berlin, 1912.

[Hof89] C. M. Hoffmann. *Geometric and Solid Modeling.* Morgan Kaufmann Publishers, Inc., 1989.

[Hon93] H. Hong. Efficient method for analyzing topology of plane real algebraic curves. Technical Report 93-21, RISC-Linz, J. Kepler Universität, Linz, Austria, April 1993.

[Hur12] W. A. Hurwitz. On the pseudo-resolvent to the kernel of an integral equation. *Trans. Amer. Math. Soc.*, 13:405–418, 1912.

[IK66] E. Isaacson and H.B. Keller. *Analysis of Numerical Method.* John Wiley & Sons, New York, 1966.

[Joh91] J. R. Johnson. *Algorithms for Polynomial Real Root Isolation.* PhD thesis, The Ohio State University, 1991.

[KM95] S. Krishnan and D. Manocha. Numeric-Symbolic Algorithms for Evaluating One-Dimensional Algebraic Sets. In A. H. M. Levelt, editor, *Proceedings of ISSAC-95*, pages 59–67, Montreal, Canada, 1995.

[KP91a] D. J. Kriegman and J. Ponce. A new curve tracing algorithm and some applications. In P. J. Laurent, Le Méhauté, and L. Schumaker, editors, *Curves and Surfaces*, 1991.

[KP91b] G. A. Kriezis and N. M. Patrikalakis. Rational polynomial surface intersections. In *Proc. 17th ASME Design Automation Conf.*, 1991.

[KPP90] G. A. Kriezis, P. V. Prakas, and N. M. Patrikalakis. A method for intersecting algebraic surfaces with rational polynomial patches. *Computer Aided Design*, 22(10):645–654, 1990.

[KPPW92] G. A. Kriezis, P. V. Prakas, N. M. Patrikalakis, and F. E. Wolter. Topological and differential equation methods for surface intersection. *Computer Aided Design*, 24(1):41–55, 1992.

[KS95] D. Kapur and T. Saxena. Comparison of various multivariate resultant formulations. In *The proceeding of ISSAC'95*, 1995.

[KSY94] D. Kapur, T. Saxena, and L. Yang. Algebraic and geometric reasoning using Dixon resultants. In *The proceeding of ISSAC'94*, 1994.

[Loo82] R. Loos. Computing in Algebraic Extensions. In B. Buchberger, G. E. Collins, and R. Loos, editors, *Computer Algebra*, pages 173–187. Springer Verlag, New York, 1982.

[Moo20] E. H. Moore. On the reciprocal of the general algebraic matrix (abstract). *Bull. Amer. Math. Soc.*, 26:394–395, 1920.

[Moo79] R. E. Moore, editor. *Methods and Applications of Interval Arithmetic.* Studies in Applied Mathematics, SIAM, 1979.

[Mor87] A. Morgan. *Solving polynomial systems using continuation for engineering and scientific problems.* Prentice-Hall, Inc., 1987.

[MRR88] R. Mines, F. Richman, and W. Ruitenberg. *A Course on Constructive Algebra.* Springer-Verlag, New York-Berlin-Heidelberg, 1988.

[Nie93] G. M. Nielson. Cagd's top ten: What to watch. *IEEE Computer Graphics & Applications*, Jan 1993.

[Pat92] N. M. Patrikalakis. Interrogation of surface intersections. In R. E. Barnhill, editor, *Geometry Processing for Design and Manufactoring*, pages 161–185. SIAM, Philadelphia, 1992.

[Pat93] N. M. Patrikalakis. Surface-to-surface intersections. *IEEE Computer Graphics & Applications*, 1(1):89–95, 1993.

[Pen55] R. Penrose. A generalized inverse for matrices. *Proc. Cambridge Philos. Soc.*, 51:406–413, 1955.

[PM91] M.H. Protter and C.B. Morrey. *A First Course in Real Analysis.* Springer, New York, 1991.

[PP90] N. M. Patrikalakis and P. V. Prakash. Surface intersections for geometric modeling. *J. Mechanical Design*, 112(1):100–107, 1990.

[Rei68] W. T. Reid. Generalized inverses of differential and integral operators. In *Proceedings of the Symposium on Theory and Applications of Generalized Inverses of Matrices*, pages 1–25. Texas Tech. Press, Lobbock, Texas, 1968.

[Rit32] J. F. Ritt. *Differential Equations from the Algebraic Standpoint.* Amer. Math. Soc., New York, 1932.

[Rit50] J. F. Ritt. *Differential Algebra*, volume 33. AMS Colloquium Publications, New York, 1950.

[Sei74] A. Seidenberg. Constructions in Algebra. *Trans. Am. Math. Soc.*, 197: pages 273–313, 1974.

[Sei84] A. Seidenberg. On the Lasker–Noether decomposition theorem. *Am. J. Math.*, 106:611–638, 1984.

[Tau94a] Gabriel Taubin. Distance approximations for rasterizing implicit curves. *ACM Transactions on Graphics*, 13(1):3–42, 1994.

[Tau94b] Gabriel Taubin. Rasterizing algebraic curves and surfaces. *IEEE Computer Graphics and Applications*, March 1994.

[Tra92] Quoc Nam Tran. An algebraic approach to mechanical theorem proving in geometry. Master's thesis, The Asian Institute of Technology, Bangkok, Thailand, 1992.

[Tra94a] Quoc Nam Tran. Extended Newton's method for finding the roots of an arbitrary system of equations and its applications. Technical Report 94–49, RISC-Linz, J. Kepler Universität, Linz, Austria, 1994.

[Tra94b] Quoc Nam Tran. Extended Newton's method for finding the roots of an arbitrary system of equations and its applications. In M. H. Hamza, editor, *Proceedings of the Twelfth IASTED International Conference: Applied Informatics*. IASTED, 1994.

[Tra95a] Quoc Nam Tran. A hybrid symbolic-numerical method for tracing plane and space algebraic curves. In *11th European Workshop on Computational Geometry CG'95*, Linz, Austria, 1995. RISC-Linz.

[Tra95b] Quoc Nam Tran. A hybrid symbolic-numerical method for tracing surface-to-surface intersections. In A. H. M. Levelt, editor, *Proceedings of ISSAC-95*, Montreal, Canada, 1995. ACM.

[Tra95c] Quoc Nam Tran. On the symbolic-numerical methods for finding the roots of an arbitrary system of non-linear algebraic equations. In *Proceedings of the First Asian Technology Conference in Mathematics, ATCM-95*, Singapore, 1995.

[Tra96a] Quoc Nam Tran. Extending newton's method for finding the roots of an arbitrary system of equations and its applications. *International Journal of Modeling and Simulation*, 1996. to appear.

[Tra96b] Quoc Nam Tran. A hybrid symbolic-numerical method for tracing algebraic curves in higher dimensions. In *ICCAM–96 conference, Leuven, Belgium*, 1996. preprint.

[Tra96c] Quoc Nam Tran. A note on irreducible decomposition of algebraic sets and automatic theorem proving. Submitted to the special issue of the Journal of Automatic Reasoning on Computer Algebra and Automatic Theorem Proving, 1996.

[Usp48] J.V. Uspensky. *Theory of Equations.* McGraw-Hill Book Company, Inc., Cambridge, 1948.

[W$^+$91] Y. Wang et al. Intersection of parametric surfaces for next generation geometric modeling systems. In J. Turner, J. Pegna, and M. Wozny, editors, *Product Modeling for Computer Aided Design and Manufacturing*, pages 75–96. Elsevier, 1991.

[Wal93] Bernhard Wall. *Symbolic Computation with Algebraic Sets.* PhD thesis, RISC-Linz, Universität Linz, 1993.

[Wan92] D. Wang. Irreducible Decomposition of Algebraic Varieties via Characteristic Sets and Gröbner Bases. *Computer Aided Geom. Design*, 9: pages 471–484, 1992.

[Wan94] D. Wang. Characteristic Sets and Zero Structure of Polynomial Sets. *A lecture note at RISC-Linz*, 1994.

[Win93] Franz Winkler. Algebraic computation in geometry. In G. Jacobs, N. E. Oussous, and S. Steinberg, editors, *IMACS'93 Conference, Lille, special issue J. Math. in Comp. Simulation*, 1993.

[Win96] Franz Winkler. *Polynomial Algorithms in Computer Algebra.* Texts and Monographs in Symbolic Computation. Springer-Verlag, 1996.

[Wu84] W.T. Wu. Basic Principles of Mechanical Theorem Proving in Elementary Geometries. *Journal Sys. Sci. & Math. Scis.*, 4: pages 207–235, 1984.

[Wu94] W.T. Wu. *Basic Principles of Mechanical Theorem Proving in Geometries.* Springer-Verlag, 1994. Translated from Chinese by Xiaofan Jin and Dongming Wang.

Index

Trần Quốc–Nam

Research Institute for Symbolic Computation
RISC–Linz, Johannes Kepler University
A–4040 Linz, Austria
E-mail: tqnam@risc.uni-linz.ac.at

Personal

Born June 22, 1963, Ho Chi Minh city, Vietnam.

Education

Ph.D. studies, Computer Science and Symbolic Computation. Research Institute for Symbolic Computation (RISC–Linz), Johannes Kepler University, Linz, Austria.

M.Sc., Computer Science, 1992. Division of Computer Science, Asian Institute of Technology, Bangkok, Thailand.

Dipl-Ing., Mathematics and Computer Science, 1984. Faculty of Mathematics & Computer Science, State University at Ho-Chi-Minh city, Vietnam.

Positions

1995–now Project's main developer, Project HySaX. Research Institute for Symbolic Computation.

1994–1995 Research Associate, Project POSSO. Research Institute for Symbolic Computation.

1993–1994 Research Associate, Project SACLIB. Research Institute for Symbolic Computation.

1993–1993 Research Associate, Project Robotic. Research Institute for Symbolic Computation.

1992–1993 Research Associate, Asian Institute of Technology.

1992–1992 Teaching Assistant, Asian Institute of Technology.

1985–1990 Consultant at the Computer Center of the Ministry of Industries, Vietnam.

Affiliations

Member of ACM

Awards and Honors

1993 Scholarship for Ph.D. studies at Research Institute for Symbolic Computation.

1991 Scholarship from the government of Australia for M.Sc. studies at the Asian Institute of Technology.

1978 An award for excellent studies at the high school.

Publications

Tran93 Tran, Quoc Nam and Kanchanasut, K. An Efficient Algorithm for Computing a Triangular Form. Technical Report, Asian Institute of Technology, Bangkok, Thailand, 1993.

Tran94a Tran Quoc-Nam. Extending Newton's Method for Finding the Roots of an Arbitrary System of Equations and Its Applications. In Hamza, M. H., editor, *Proceedings of the Twelfth IASTED International Conference: Applied Informatics*, 1994.

Tran94b Tran Quoc-Nam. On the Extended Newton's Method for Finding the Roots of an Arbitrary System of Equations. RISC–Linz Technical Report No. 94–49, 1994

Tran95a Tran Quoc-Nam. A Hybrid Symbolic-Numerical Method for Tracing Surface-to-Surface Intersections. In Levelt, A. H. M., editor, *Proceedings of International Symposium on Symbolic and Algebraic Computation (ISSAC-95)*, Montreal, Canada, 1995.

Tran95b Tran Quoc-Nam. A Hybrid Symbolic-Numerical Method for Tracing Plane and Space Algebraic Curves. *the 11th European Workshop on Computational Geometry CG'95*, Linz, Austria, 1995.

Tran95c Tran Quoc-Nam. On the Symbolic-Numerical Methods for Finding the Roots of an Arbitrary System of Non-linear Algebraic Equations. *Proceedings of the First Asian Technology Conference in Mathematics (ATCM-95)*, Singapore, 1995.

Tran96a Tran Quoc-Nam. Extending Newton's Method for Finding the Roots of an Arbitrary System of Equations and Its Applications. *International Journal of Modeling and Simulation*, 1996. To appear.

Tran96b Tran Quoc-Nam. A Note on Irreducible Decomposition of Algebraic Sets and Automatic Theorem Proving. Preprint, 1996.

Tran96c Tran Quoc-Nam. A Hybrid Symbolic-Numerical Method for Tracing Algebraic Curves in Higher Dimensions. *ICCAM–96 conference*, Leuven, Belgium. International Congress on Computational and Applied Mathematics.

Tran96d Tran Quoc-Nam. Intersection of Surfaces. *The Seventh Mathematics of Surfaces, Dundee, U. K.* The Institute of Mathematics and its Applications.

Tran96f Tran Quoc-Nam. Extending Dixon's Resultant Using Pseudoinverse Matrices. To appear in the book "Automated Deduction in Geometry", 1996.

Personal Status

Married since 1985.

Two children.