Symbolic and Numerical Computation for Artificial Intelligence

COMPUTATIONAL MATHEMATICS AND APPLICATIONS

Series Editors

J. R. WHITEMAN
Institute of Computational Mathematics, Brunel University, England

and J. H. DAVENPORT
School of Mathematical Sciences, University of Bath, England

E. HINTON and D. R. J. OWEN: Finite Element Programming
M. A. JASWON and G. T. SYMM: Integral Equation Methods in Potential Theory and Elastostatics
J. R. CASH: Stable Recursions; with applications to the numerical solution of stiff systems
H. ENGELS: Numerical Quadrature and Cubature
L. M. DELVES and T. L. FREEMAN: Analysis of Global Expansion Methods: weakly asymptotically diagonal systems
J. E. AKIN: Application and Implementation of Finite Element Methods
J. T. MARTI: Introduction to Sobolev Spaces and Finite Element Solution of Elliptic Boundary Value Problems
H. R. SCHWARZ: Finite Element Methods
J. DELLA DORA and J. FITCH: Computer Algebra and Parallelism
E. TOURNIER: Computer Algebra and Differential Equations
A. ŽENÍŠEK: Nonlinear Elliptic and Evolution Problems and Their Finite Element Approximations
M. SINGER: Differential Equations and Computer Algebra
G. CHEN and J. ZHOU: Boundary Element Methods
B. R. DONALD, D. KAPUR and J. L. MUNDY: Symbolic and Numerical Computation for Artificial Intelligence

Symbolic and Numerical Computation for Artificial Intelligence

edited by

Bruce Randall Donald
Department of Computer Science
Cornell University, USA

Deepak Kapur
Department of Computer Science
State University of New York, USA

Joseph L. Mundy
AI Laboratory
GE Corporate R&D, Schenectady, USA

Academic Press
Harcourt Brace Jovanovich, Publishers
London San Diego New York
Boston Sydney Tokyo Toronto

ACADEMIC PRESS LIMITED
24–28 Oval Road
London NW1

US edition published by
ACADEMIC PRESS INC.
San Diego, CA 92101

This book is printed on acid-free paper

A catalogue record for this book is available from the British Library

ISBN 0-12-220535-9

Printed and Bound in Great Britain by
The University Press, Cambridge

Contents

Contributors

Messaoud Benantar, *Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA*

Rupak Biswas, *Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA*

David B. Cooper, *Laboratory for Engineering Man/Machine Systems, Division of Engineering, Brown University, Providence, RI 02912, USA*

James H. Davenport, *School of Mathematical Sciences, University of Bath, Claverton Down, Bath BA2 7AY, UK*

Michael C. Dewar, *School of Mathematical Sciences, University of Bath, Claverton Down, Bath BA2 7AY, UK*

Bruce R. Donald, *Department of Computer Science and Center for Applied Mathematics, Cornell University, Ithaca, NY 14853, USA*

Joseph E. Flaherty, *Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA*

David Forsyth, *Department of Computer Science, University of Iowa, Iowa, IA 52242, USA*

Elias N. Houstis, *Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA*

Daniel P. Huttenlocher, *Computer Science Department, Cornell University, Ithaca, NY 14853, USA*

Deepak Kapur, *Institute for Programming and Logics, Department of Computer Science, State University of New York, Albany, NY 12222, USA*

Klara Kedem, *Computer Science Department, Tel Aviv University, Tel Aviv, Israel*

David J. Kriegman, *Center for Systems Science, Department of Electrical Engineering, Yale University, New Haven, CT 06520, USA*

Yagiti N. Lakshman, *Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, USA*

Catherine Lassez, *IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA*

Jean-Louis Lassez, *IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA*

Philip S. Milne, *School of Mathematical Sciences, University of Bath, Claverton Down, Bath BA2 7AY, UK*

Peter K. Moore, *Department of Mathematics, Tulane University, New Orleans, Louisiana 70118, USA*

Alexander P. Morgan, *Mathematics Department, General Motors Research Laboratories, Warren, Michigan 48090, USA*

Joseph L. Mundy, *Artificial Intelligence Laboratory, GE Corporate Research and Development, Schenectady, NY 12345, USA*

Kazuo Murota, *Research Institute for Mathematical Sciences, Kyoto University, Kyoto 606, Japan*

Dinesh K. Pai, *Department of Computer Science, University of British Columbia, Vancouver V6T 1Z2, Canada*

Jean Ponce, *Beckman Institute, Department of Computer Science, University of Illinois, Urbana, IL 61801, USA*

John R. Rice, *Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA*

Michael G. Richardson, *Numerical Algorithms Group Ltd., Wilkinson House, Jordan Hill Road, Oxford OX2 8DR, UK*

Charles Rothwell, *Robotics Research Group, Department of Engineering Science, Oxford University, Oxford OX1 3PZ, UK*

Stanly Steinberg, *Department of Mathematics and Statistics, University of New Mexico, Albuquerque, New Mexico 87131, USA*

Gabriel Taubin, *Exploratory Computer Vision Group, IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA*

Sanjiva Weerawarana, *Department of Computer Sciences, Purdue University, West Lafayette, IN 47907, USA*

Richard Zippel, *Department of Computer Science, Cornell University, Ithaca, NY 14853, USA*

Andrew Zisserman, *Robotics Research Group, Department of Engineering Science, Oxford University, Oxford OX1 3PZ, UK*

Preface

The papers in this volume are based on talks given at a workshop on the *Integration of Numerical and Symbolic Computing Methods,* held in Saratoga Springs, New York, in July 1990. The workshop was sponsored by the National Science Foundation (NSF), the Air Force Office of Sponsored Research (AFOSR), General Electric Research and Development, Schenectady, and the State University of New York at Albany. Over forty researchers from industry, academia and government participated in the workshop.

Papers included in the book were selected after a review of the revised papers submitted for the proceedings. We are grateful to the workshop sponsors, participants, and referees for their hard work. We especially thank Dr. Abe Waksman of the AFOSR and Dr. Kamal Abdali of the NSF for providing us the necessary funding and Sally Goodall for making excellent arrangements for the workshop. We would like to thank Debbie Lee Smith for her assistance in coordinating the refereeing of the manuscripts. Finally, we are grateful to Roli Varma but for whose careful and diligent work on proof reading and editing the papers to satisfy the publisher's formatting requirements, this book would have not come out.

Bruce Randall Donald

Deepak Kapur

Joseph L. Mundy

Introduction

1. Background

Engineering and applied mathematics face the challenge of exploiting theoretical results for practical applications. Real-world engineering problems are often not amenable to exact solutions. For example, in engineering design, one often sees the following scenario. First, we use theoretical methods to work out an abstract or idealized model. This stage derives useful relations to guide the design process. Next, we often resort to approximate, numerical methods for solving specific problems. The modeling process is iterative in nature; symbolic techniques are used to study properties of an idealized model; the model is then refined or fine-tuned to bring it closer to a real-world phenomenon.

Over the last decade, there has been considerable progress in investigating methods of symbolic mathematics in many application areas of computer science and artificial intelligence, such as engineering design, solid and geometric modeling, robotics and motion planning, and machine vision. Most of this research, however, has been theoretical, and its application to real problems has been limited because of the difficulty of applying the combinatorially expensive symbolic techniques. In addition, there are few well-understood mechanisms in either computer algebra or other symbolic disciplines for handling data error or limited computational precision. Typically, in practice, such problems are addressed by using numerical methods to approximate the underlying mathematical models. There is a need to investigate approaches for systematic integration of symbolic techniques with numerical techniques. Symbolic methods can provide insight into the structure and stability of problem solutions, while numerical methods are efficient for handling the huge amount of data that arises in practical applications.

Numerical methods have proven successful in solving many practical problems, but their use requires considerable sophistication. It is difficult to characterize the domain where such methods will converge or to predict the accuracy that can be achieved. A symbolic package as a front end to a numerical package can act as a kind of expert system to help a user to properly set up a problem for numerical computing. For example, in finite-element methods, symbolic methods which combine geometric and physical theories can be used to help determine effective finite element mesh structures for numerical relaxation techniques.

Symbolic methods such as those supported in systems like MAPLE, MACSYMA and MATHEMATICA provide mechanisms for manipulating the analytic structure of physical problems. For example, simple systems of equations can be solved using elimination methods. It is often possible to solve integrals in closed form or to use rapidly convergent series to evaluate specific cases. Another key application is convexity analysis. For specific classes of problems, convexity determines the regions in which stable and unique solutions can be obtained.

The problem of finding real solutions of a system of polynomial equations arises in almost all the applications mentioned above. While there has been a great deal of progress

in the development of theoretically efficient algorithms for the solution of such systems, these algorithms have not achieved widespread use in engineering applications. In an engineering system, one typically uses numerical root-tracking methods which have the disadvantage of being combinatorially imprecise. *Homotopy* methods are a promising example for integration of symbolic and numerical methods.

In the following, we give an overview of some applications in machine vision, robotics, and engineering design where there is a need for integrating symbolic and numerical methods. This list is by no means exhaustive, but we believe it is a good representative of applications in these domains. To give a flavor for how symbolic and numerical techniques can be combined to obtain practical, efficient solutions to engineering problems, we discuss a case study–the problem of computing approximate rational coefficients in a rotational transformation that do not introduce any scaling. This problem comes up in the implementation of many geometric algorithms. This is followed by an overview of the papers in this volume.

2. Machine Vision

Machine vision exemplifies the interaction between symbolic and numerical methods. In *model-based* vision, the input data is a two-dimensional array of numbers representing image intensity or surface range, along with some description of models in the form of geometric and topological relationships. The former encodes the sensory input, the latter, the *model*. The model represents geometric assumptions and expectations about the world the sensors interrogate. The goal is to extract conceptual descriptions of the world from varying sensory data.

2.1. Constraint-based Object Modeling

Recent developments in constraint-based object modeling represent a direct integration of symbolic and numerical methods to accomplish this task. Objects are specified in terms of geometric constraints. Any instance of an object must obey these constraints. An actual instance of the object is found by minimizing the "match error". This error measures the "difference" between the object instance (which is defined by the system of model constraints) and the empirical feature measurements taken from the image data. The error minimization is achieved by adjusting free parameters of the constraint system. Elimination methods such as the characteristic set algorithm of Ritt and Wu, and Buchberger's Gröbner basis algorithm, can be useful in eliminating dependent parameters and for determining regions of stable convergence for the constraint system. Nonlinear programming and optimization techniques are useful for minimizing error between predicted data and observational data.

2.2. Model-based Vision, Parameterized Modeling and Matching

The use of elimination theory in machine vision has been demonstrated in Sugihara (1984) and Cyrluk *et al.* (1987). Earlier work focussed on polyhedral objects, in which different idealized images of objects were matched for consistency. The consistency check was performed using algebraic methods; for example, there are many numerical methods for solving linear constraints, and the Gröbner basis algorithm can be used for nonlinear constraints.

More recently, in the work of Ponce and Kriegman (in this volume), the idea is to represent a curved object, such as a toroid, as an algebraic surface, e.g. a rational bicubic. Given polynomials describing the surface and polynomials describing the viewing geometry, we form a "combined system". The visual contours of the surface can be determined by eliminating surface parameters from this combined system. This visual contour is represented as an algebraic curve; that is, the curve is defined by the occluding contours derived by simultaneous solution of the combined system of surface and viewing equations. Next, variables are eliminated from the combined system using resultants. Then, the discontinuities in the image mapping are determined numerically, by finding the roots of the resulting system.

2.3. Geometric Invariants and Matching

Another example from vision entails the use of algebraic invariant theory to define invariant numerical methods for approximating data with algebraic curves and surfaces. This approach may, indeed, be more generally applicable to other data modeling problems. The idea is that the curve or surface fitting procedure should be invariant under the group of coordinate transformations (e.g. rotation, translation, scaling and perspective mappings). The use of algebraic invariants as image features for matching, and to define landmarks for robot navigation, has already been explored by Nielson (1988) and Weiss (1989). In order for these invariants to be reliably extracted from image data, the fitting process itself must be invariant under the allowed group of image transformations. There exists a powerful symbolic mechanism for generating algebraic invariants for polynomial curves and surfaces, which was developed during the last century by Young. An interesting new avenue of research is emerging which employs these invariants as metrics in curve and surface fitting procedures (Bookstein, 1979; Kapur and Mundy, 1989). Of crucial importance is the development of techniques for establishing the existence and uniqueness properties of such fitting methods.

The output of such a fitting procedure is an algebraic curve or an algebraic surface, whose invariant properties are used to quickly search a given data base of models; the algorithm uses invariants as indexing functions.

3. Engineering Design and Robotics

There are many computational problems in robotics in which combined numerical and algebraic techniques can leverage efficient solutions that are relevant to engineering. Many algebraic techniques for problems such as kinematic motion planning are not yet efficient in practice. Their efficiency can be improved by the use of numerical methods. Many robotics problems have two components: a "dynamical systems" component in which one attempts to predict the long-term behavior of some system whose dynamics is specified by differential and algebraic constraints; and a "combinatorial" component which arises because, in general, there are a finite number of holonomic constraints in the environment. Often, the dynamical systems problems are amenable to numerical techniques (in fact, there may exist no algebraic solutions), while the combinatorial component can be attacked from a careful computational-geometric analysis of the holonomic constraints.

3.1. Robot Motion Planning with Dynamics

In computational geometry, it is often hoped that by making simplifying assumptions in robotics problems, precise algorithmic solutions could be obtained using algebraic methods, and that these solutions could then be generalized to "real" problems. A major stumbling block to generalizing these results was the fact that many kinematic motion planning problems are algebraic, whereas the introduction of rigid body dynamics leads to non-algebraic solution trajectories. These solutions are perhaps more amenable to a numerical attack. Methods in control theory and optimal control tend to involve numerical techniques, with an optimization flavor, whereas work in motion planning tends to involve symbolic or algebraic techniques, with a combinatorial flavor. Many robotics and engineering applications seek to link these fields, combining results from control and complexity, from optimization and combinatorial analysis, and from numerical and symbolic methods. A successful integration and application of algebraic and numerical methods has been demonstrated in designing good near-optimal control for robot motion planning (Canny *et al.*, 1988; Donald and Xavier, 1989) and for computing forward projections of a dynamical system.

3.2. Design for Assembly

A key issue in design concerns the ability of a computer system to predict the behavior of a hypothetical object from the built environment. For example, the object might be a CAD model of a mechanism. We may wish to simulate this object to determine whether it has the desired function or behavior. More importantly, based on the results of the simulation, the design system may make inferences from the observed performance, and modify the design to improve it. Hence, the dynamics that determine the interaction of objects with the world must be made explicit and computational.

Algebraic methods from robot motion planning can be combined with numerical methods from dynamical systems and finite-element analysis to develop algorithms that can analyze and generate designs for objects so that they will be easy to assemble. In particular, real objects that robots might assemble are typically not rigid. A systematic program for reasoning about and predicting their motions in contact is needed. It is possible to model the physics of interaction between the flexible parts and the environment (including their mating parts). Combinatorially precise algorithms can be developed for predicting the motion of a flexible object near and in contact with its mating part. Donald and Pai (1990) explored the use of the analysis algorithm in an approach to *"design as search"*, in which they modify and improve an existing design by changing its geometry incrementally and applying the analysis from the motion prediction algorithm.

3.2.1. Differential Theories of Mechanics

It is possible to view the motion prediction problem with rotational compliance and quasi-static mechanics as a problem that can be solved by careful reduction to the intersection, or collision detection problems (Canny, 1986; Donald, 1987). Quasi-static analysis is a differential theory of mechanics, which can be integrated to predict the long-term behavior. When a theory is exact and closed-form integrable, combinatorially precise algorithms for predicting and planning the motion of objects have been developed (Erdmann, 1986; Briggs, 1989; Brost and Mason, 1989; Canny, 1989; Donald, 1989, 1990).

When a closed-form solution is not known, numerical integration methods must be used to compute solutions in some cases. A theory of mechanics with rotational compliance that is closed form *algebraic* has been developed by Donald and Pai (1990). The solutions paths are parameterized by time and are piecewise quadratic or linear. It is possible to generalize the techniques discussed there to encompass certain simple types of uncertainty in control and initial conditions. In principle, these algorithms must be implemented using exact-precision, algebraic numbers. In practice, we use finite-precision approximation techniques. Robustness is a key issue, and algorithms that are theoretically correct with exact precision are often numerically unstable. For including continuously deformable objects, one approach would involve combining numerical finite-element methods with the algebraic theories of mechanics.

4. A Case Study

Integration of numerical and symbolic techniques in a single system can be exceedingly difficult. It is necessary to review the basic ideas in numerical and symbolic computation, and how they are used in applications that require their integration. To this end, in our introduction, we wish to illustrate the kind of issues that arise when we integrate numerical and symbolic computing. As an example, we examine a very special case—the problem of implementing robust geometric algorithms using exact, and inexact arithmetic. In this example, we first develop the fundamental theory (of rational sine approximations) that is adequate for this situation. Then, we generalize our approach to derive better algorithms by a careful bit-complexity analysis. Finally, we hint at the many practical issues and new directions.† Once again, numerical and symbolic techniques will join forces to provide a solution.

For this simple problem, we find that numerical and symbolic techniques with a very simple structure suffice. This permits us to illustrate by a specific example the situation that the articles in this volume later investigate more generally, in detail, and for a wide range of applications.

4.1. Geometric Algorithms

Many geometric algorithms can be implemented using exact-precision, algebraic numbers. In practice, we might use finite-precision approximation techniques. Robustness is a key issue, and algorithms that are theoretically correct with exact precision are often numerically unstable. A major component of design research consists of building a system that can strengthen the theoretical algorithms (e.g. by adding consistency checks) to make them practical.

Algorithms in computational geometry often use the real-RAM model of computation. In particular, this model assumes that exact real numbers can be stored in and retrieved from memory in constant $O(1)$ time, and that field operations $(+, -, *, /)$ and certain other operations (like square root, sine and cosine) are also "exact", and can be applied in constant time. This assumption makes theoretical computational geometry algorithms, even well-understood algorithms like plane-sweep for polygon union

† John Canny told Bruce Donald about these methods at the Saratoga Springs Workshop. Donald then told his *CS 661 (Robotics)* class at Cornell University. This case study is based on notes developed by Gene Ressler, and on the paper by Canny *et al.* (1992).

(Preparata and Shamos, 1985), difficult to implement and numerically unstable. These algorithms obtain good combinatorial complexity bounds by exploiting *ordering* properties of the edges, vertices and intersections. Floating point implementations are subject to numerical problems; for example, the "symbolic" ordering properties and the "numerical" ordering properties do not agree, and the algorithms fail. This is not a "mere engineering difficulty"—it prevents these algorithms from being used in practice. One possible solution is to use *rational arithmetic*, which is exact. For certain algorithms (e.g. plane-sweep), one can show that in a careful implementation, the "size" (number of bits) of the rational numbers at most doubles. ‡

However, many applications from scientific computation and artificial intelligence require that these algorithms be mixed, or interwoven with operations that destroy the exactness of the rational representation. One such operation is rotation, which is the primary concern of this case study. For example, suppose we have two sets of polygons, A and B, that model rigid objects in a 2D world. Suppose we *rotate* A by θ radians (in the plane), to obtain $A(\theta)$; i.e. for each vertex (x, y) of A we compute a new one (x', y') by the familiar transformation:

$$\begin{aligned} x' &= x \cdot \cos\theta - y \cdot \sin\theta \\ y' &= x \cdot \sin\theta + y \cdot \cos\theta \end{aligned} \tag{4.1}$$

This models a physical rotation of the object modeled by A with respect to B, but here a problem arises. For almost all θ, one of $\sin\theta$ and $\cos\theta$ is irrational. Arbitrary irrationals cannot be represented in computing machines. We will have to approximate somewhere, but where?

One idea is simply to approximate the irrational sines and cosines in (4.1) with "nearby" rationals. However the resulting transformation is typically no longer orthonormal — no longer a rotation, but, instead, a rotation and scaling. In our example, instead of $A(\theta)$ we obtain $(1+\delta_s)\cdot A(\theta+\delta\theta)$, where δ_θ and δ_s are small constants. This has two distinguishable effects. First, the rotated versions of A are not rotated exactly the desired amount. We regard this as a good kind of approximation because it still models a nearby configuration of A and B in the world. Second, the rotated polygons of A have changed size with respect to those of B. This is a bad approximation in that it is inconsistent with A and B as models of rigid objects.

Hence the thrust of this case study is to find rational rotation coefficients for some angle close to the one desired that have small representations and do not introduce scaling (i.e. we want $\delta_s = 0$). We call these *pure rotations* and give efficient algorithms to find them. More precisely, for given angle θ and tolerance ϵ_θ, our final algorithm returns a rational S with the following properties:

1 $\delta_\theta = |\sin^{-1} S - \theta| < \epsilon_\theta$

2 The corresponding cosine, $\sqrt{1-S^2}$, is also rational.

3 S has at most one bit more than the shortest rational satisfying 1 and 2.

For our purposes, the *length* of a rational is the magnitude of its denominator. Thus, rational P is *shorter* (resp. *longer*) than rational Q if P's denominator has smaller (resp. greater) magnitude. The *number of bits* of a rational is the number of bits in its denominator not counting leading zeros. Shortness of sines is vital for applications as, in

‡ For other work on robust geometric algorithms see, for example, Fortune and Milenkovic (1991), Hoffmann *et al.* (1988) and Li and Milenkovic (1990).

Specification: Rational sine
Inputs: Angle θ, tolerance ϵ_θ.
Output: Rational sine S such that:
(a) $|\sin^{-1} S - \theta| < \epsilon_\theta$
(b) S is as short as possible.

Figure 1. What is wanted is an algorithm with this specification.

general, the number of bits in the rotation coefficients is added to the bits in each point in obtaining the rotated points.

Our algorithms are iterative and terminate in $O(n)$ iterations where $n = \log(1/\epsilon_\theta)$. Each iteration requires $O(M(n))$ time where $M(n)$ is the time to multiply two rationals of n bits. With Schönhage-Strassen multiplication, this yields overall complexity of $O(n^2 \log n \log\log n)$. Though it is practical to implement the algorithm entirely with rational arithmetic and thus achieve arbitrary precision, we concentrate on simple codes that use double precision floating point arithmetic to compute error terms. These perform well for $\epsilon_\theta \geq 10^{-10}$ and thus may be regarded as constant time operations. In practice, they run in a few milliseconds.

A secondary result of this case study concerns the variant of Euclid's algorithm often used to approximate arbitrary numbers with nearby rationals to within given ϵ. An example is the `rationalize` function of Common Lisp implementations (where ϵ is the machine floating point precision). We show this algorithm does not always return the shortest possible rational and give a fix so that it does. The final algorithm is short and simple, belying the rich structure of the problem it solves the assortment of techniques available to attack it. As such, we chronicle various approaches that lead to the final results. In section 4.2, we couch the problem in convenient terms. In section 4.3, we describe a brute force approach that yields a few rational sines. Section 4.4 gives our first iterative algorithm, with encouraging insights. Section 4.5 applies well-known results on Diophantine equations to show that finding sines is equivalent to finding rational approximations to arbitrary numbers. Section 4.6 shows a variant of Euclid's algorithm to perform this approximation. We can analyze the Diophantine technique and Euclid's algorithm, and derive an algorithm that satisfies the claims above.

4.2. Setting Up the Problem

Let $s = \sin\theta$ and $c = \cos\theta$, then we are interested in rational solutions to:

$$s^2 + c^2 = 1. \tag{4.2}$$

Picking such a rational solution is clearly equivalent to picking θ such that $\sin\theta$ and $\cos\theta$ are both rational, but as mentioned above, such θ values are often irrational, so we do not have the freedom to compute them directly; we must be more subtle.

Define a *rational sine* to be any rational solution for s in (4.2). An alternative definition will also be helpful: let $\Omega(x) = \sqrt{1 - x^2}$. Then a rational sine is any rational number S such that $\Omega(S)$ is also rational.† What is wanted is an algorithm with the specification in figure 1.

† Of course $\Omega(S)$ is also a rational sine, as $\Omega(\Omega(S)) = S$.

0 – 0	10 – 92/533	20 – 51/149	30 – 451/901	40 – 88/137
1 – 115/6613	11 – 93/485	21 – 135/377	31 – 180/349	41 – 48/73
2 – 57/1625	12 – 76/365	22 – 372/997	32 – 28/53	42 – 65/97
3 – 39/761	13 – 156/685	23 – 348/877	33 – 432/793	43 – 429/629
4 – 29/421	14 – 205/853	24 – 231/569	34 – 161/289	44 – 555/797
5 – 23/265	15 – 69/269	25 – 36/85	35 – 228/397	45 – 697/985
6 – 19/181	16 – 7/25	26 – 39/89	36 – 504/865	
7 – 32/257	17 – 120/409	27 – 300/661	37 – 3/5	
8 – 129/929	18 – 57/185	28 – 8/17	38 – 580/941	
9 – 100/629	19 – 12/37	29 – 189/389	39 – 341/541	

Figure 2. Short rational sines for 0 to 45 degrees.

4.3. Simple-minded Search

Our first attempt is purely pragmatic. In implementing planners for paths of robots that rotate, one approach is to consider only a finite, uniformly spaced set of rotation angles in $[0, 2\pi)$. The first job of the planner is then to compute configuration space obstacles for this set of angles as described above in section 4.8.1. For this purpose, it appears natural to see if there are enough short rational sines to fill a table with reasonable density, perhaps one per degree. Considering the equation

$$\left(\frac{a}{b}\right)^2 + \left(\frac{c}{d}\right)^2 = 1 \quad \Rightarrow \quad b^2 - a^2 = \left(\frac{bc}{d}\right)^2$$

it is not hard to see that we are looking for all pairs of integers (a, b) such that $a^2 - b^2$ is a perfect square. The pragmatic approach is to search exhaustively for all of these with fewer than some small number of digits. It is surprising to find 159 pairs where $0 \leq b \leq a < 1000$ and $a/b < \sqrt{2}/2$, corresponding to sines for angles between 0 and $\pi/4$. Moreover, they are distributed evenly enough so that for most integral degrees, a sine is available within 0.3 degrees, and usually much closer. The exceptions are angles within 2 degrees of $90 \cdot k$ for integer k, where no 3 digit sines exist. Searching for 4 digit sines is feasible, especially to fill the empty spaces at 1 and 2 degrees, and this yields many more pairs. However, this is about the practical limit of brute force search. We have compromised on (a) of the specification to ensure (b). The result is given in figure 2. It is perhaps worth spending a minute to peruse the table. For example, note that we have $\sin 30° \approx 451/901$ when it's exactly 1/2. This is because, of course, the corresponding cosine, $\sqrt{1 - (1/2)^2} = \Omega(1/2)$ is irrational. Hence, 1/2 is not a rational sine.

4.4. Inspired Iteration

We next seek a way to iteratively refine the entries of the table to achieve arbitrarily precise results, satisfying (a) of the specification. The result is given in figure 3. The following is a brief description of the non-obvious steps. Step 5 is apparently based on the sum of sines identity $\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$ and the fact that for small k, $\sin k \cong k$. The job of step 4 then is to find a small k related to $\delta\theta$ that ensures convergence and has $\Omega(k)$ rational so that step 5 always yields a rational number. This it does. The algorithm returns reasonably short sines, about twice the optimal number

1 Retrieve S, the closest rational sine to $\sin\theta$ from a precalculated table.
2 If $|\sin^{-1} S - \theta| < \epsilon$, return S.
3 Compute a correction angle $\delta\theta = \sin^{-1} S - \theta$.
4 Compute a correction factor

$$k = \frac{2c-1}{2c^2-2c+1} \quad \text{where} \quad c = \lfloor 1/\delta\theta \rfloor.$$

5 Set $S := S\Omega(k) \pm kS$, where the sign depends on the sign of $\delta\theta$ in the obvious way. Go to step 2.

Figure 3. The first algorithm.

of bits, very quickly, but we have yielded on (b) of the specification to get (a). It remains to show that we can come quite close to achieving both at the same time.

4.5. Invoking Diophantus...

What should be noticed about the algorithm of figure 3 is that finding a suitable k, an approximation of the sine of a correction angle, is much the same as our overall requirement. If the formula in step 4 had the power to generate all possible corrections k such that $\Omega(k)$ is rational, it would also be able to generate all possible rational sines by iterating over all integers c. It is not clear what can be said about the sine-generating powers of the formula in step 4 — it was obtained *ad hoc* — but we can derive a similar form which is close to optimal for our purposes.

All solutions to the Diophantine equation

$$x^2 + y^2 = z^2$$

with y even and x, y, and z relatively prime, are of the form

$$x = m^2 - n^2 \quad y = 2mn \quad z = m^2 + n^2$$

where m and n are integers. See Canny *et al.* (1992). Rearranging, we have

$$\left(\frac{x}{z}\right)^2 + \left(\frac{y}{z}\right)^2 = 1$$

Thus all x/z and y/z are rational sines. Picking y/z because the math is a bit simpler, we have

$$\frac{y}{z} = \frac{2mn}{m^2+n^2} = \frac{2}{\frac{m}{n}+\frac{n}{m}} = \frac{2}{x+1/x}$$

where x is any rational number. Thus the expression $2/(x+1/x)$ exactly characterizes *all* rational sines.† The outline of an algorithm immediately suggests itself:

Guess rational x such that $S = 2/(x+1/x)$, satisfies $|\sin^{-1} S - \theta| < \epsilon$. Return S. (4.3)

Thus the problem is reduced to a search for x, but the search has a very specific goal. We can solve the equation $\sin\theta = 2/(x'+1/x')$ for x':

$$x' = 1/\sin\theta + \sqrt{1/\sin^2\theta - 1} \tag{4.4}$$

† The restriction of the Diophantine solutions to even y is of no concern. If some rational sine p/q has an odd numerator, then the solution with $y = 2p$, $z = 2q$ accounts for it.

$e_0, p_0, q_0, e_1, p_1, q_1 := x, 0, 1, -1, 1, 0;$
repeat
  $r := \lfloor e_0/e_1 \rfloor;$
  $e_0, p_0, q_0, e_1, p_1, q_1 := e_1, p_1, q_1, e_0 - re_1, p_0 - rp_1, q_0 - rq_1;$
until $|p_1/q_1 - x| < \epsilon$

Figure 4. Rational approximation algorithm.

Then x' is the exact value of x we need, except that it is probably irrational. It remains to approximate x' with a nearby rational number. We initially did this using bisection between $\lfloor x' \rfloor$ and $\lceil x' \rceil$, and this gave answers with one third fewer bits than the algorithm of figure 3. However, we can do better.

4.6. ...And Then Euclid

There exists an iterative algorithm to approximate any number x with a series of successively closer rationals $R_0, R_1, \ldots$. It terminates, returning x in canonical reduced form, if and only if x is rational. What makes it especially desirable for our needs is that the R_i are significantly shorter than the estimates we got by bisection; often they are the shortest possible. Figure 4 is the algorithm. The assignment statements perform all assignment of right hand side expression results to respective left hand side variables simultaneously. Variable x holds the number we are approximating. The approximations R_i are the values of p_1/q_1 in successive iterations. Iteration stops when this value is within ϵ of x. The astute reader will see embedded in this algorithm Euclid's method for finding the GCD of two integers. In effect, it is finding the "GCD" of x and 1.

The existence of this algorithm should not surprise Common Lisp users! The library function `rationalize` commonly uses it to get a rational nearby to a floating point number; e.g. `(rationalize .1111111111111111)` returns 1/9 in many implementations.

The incorporation of figure 4 and the discussion of section 4.5 yields a complete algorithm. Figure 5 shows Common Lisp code to compute rational sines in $[0, \pi/4)$. Recall that `do` loops have simultaneous assignment semantics as the notation in figure 5 below. The first line of the *do* evaluates the right hand side of (4.4) in double precision arithmetic to obtain the "exact" value of x to be approximated. Iteration proceeds as in figure 4, with the stopping criterion from (4.3). The last line computes the current most accurate value of the rational sine due to the current most accurate rational approximation of x, namely `p1/q1`.

4.7. Analysis

How short are the answers of figure 5? There are two things to consider. First, we developed this algorithm on the intuition that if we find a short x, it will indeed yield a short value of $2/(x + 1/x)$. This intuition needs verification. Second, though Euclid's algorithm is appealing, we do not know if it really produces shortest answers for a given ϵ. We deal with these matters in order.

To address the first point, suppose S^* is the shortest rational sine for angles within ϵ_θ of θ, i.e. $|\sin^{-1} S^* - \theta| < \epsilon_\theta$. We need to show the following:

Claim 4.1. *Let x be the shortest rational such that $|\sin^{-1} S - \theta| < \epsilon_\theta$, where $S = 2/(x + 1/x)$. Then S is at most one bit longer than S^*.*

```
(defun ssine (ang &optional (eps 0.01))
  (let ((s (sin ang)))
    (if (< ang eps) 0
      (do ((e0 (+ (/ 1 s) (sqrt (- (/ 1 (* s s)) 1))) e1)
           (p0 0 p1)
           (q0 1 q1)
           (e1 -1 (- e0 (* rat e1)))
           (p1 1 (- p0 (* rat p1)))
           (q1 0 (- q0 (* rat q1)))
           (rsin 1)
           rat)
          ((< (abs (- ang (asin (coerce rsin 'double-float)))) eps) rsin)
        (setq rat (truncate e0 e1)
              rsin (/ (* 2 p1 q1) (+ (* p1 p1) (* q1 q1))))))))
```

Figure 5. Code for a better rational sine algorithm.

The proof of this devolves to showing that

LEMMA 4.1. *Given p and q relatively prime,* $\gcd(2pq,\ p^2+q^2) \leq 2$.

We will not give proof here, but see Canny *et al.* (1992) for details. To summarize, this claim says that if we really *do* find the shortest x, then $2/(x+1/x)$ is either the shortest sine, or another sine only one bit longer.

That leaves the second point: does the algorithm of figure 4 produce the shortest answers? The answer is *no*, but it is quite close to one that does. To see what is going on, we need a simple tool from number theory, called *Farey sequences* (see e.g. Rademacher, 1984). The Farey sequence of order N is just the ascending sequence of canonically reduced fractions between zero and one with denominators not exceeding N. There are two fundamental theorems concerning them:

THEOREM 4.1. *Every reduced positive rational less than one appears in a Farey sequence.*

THEOREM 4.2. *Every element of the Farey sequence of order $N+1$ that is not in the sequence of order N is an N-mediant.*

An N-mediant is a number $(a+b)/(c+d)$, where a/c and b/d are adjacent elements of the order N sequence. Noting these theorems, we claim there is an obvious algorithm to find the sequence of fractions that approximate any x, $0 \leq x < 1$ in ascending order of length. We start with the order 1 sequence $0/1, 1/1$ and add the mediants closest to x in succession.

An analysis using Farey sequences allows us to at once characterize the cases where Euclid's algorithm produces non-shortest answers, and also suggests how to solve the problem. The analysis is beyond the scope of this introduction, but see Canny *et al.* (1992) for more details.

Figure 6 is the final code with separate functions to approximate numbers and give rational sines. With $\epsilon_\theta = 10^{-4}$, the sine function gives answers one to three bits shorter that figure 5 about a third of the time for random angles. Interestingly, about one in twenty returns is a bit *longer* than figure 5. This occurs when a non-shortest x generated by figure 4 happens to yield a one bit shorter sine than the shortest x. This possibility is allowed by our earlier claim, and we are thus assured that it occurs in fact.

```
;;; Find the smallest rational between x0 and x1.
(defun rat (x0 x1)
  (let ((i (ceiling x0)) (i0 (floor x0)) (i1 (ceiling x1)))
    (if (>= x1 i) i
      (do ((p0 i0 (+ p1 (* r p0)))
           (q0 1 (+ q1 (* r q0)))
           (p1 i1 p0)
           (q1 1 q0)
           (e0 (- i1 x0) e1p)
           (e1 (- x0 i0) (- e0p (* r e1p)))
           (e0p (- i1 x1) e1)
           (e1p (- x1 i0) (- e0 (* r e1))) r)
          ((<= x0 (coerce (/ p0 q0) 'double-float) x1) (/ p0 q0))
        (setq r (min (floor e0 e1) (ceiling e0p e1p)))))))

;;; Return a small rational sine for an angle in [a0,a1], 0 <= a0 < a1 < pi/2
(defun rat-sin (a0 a1)
  (if (zerop a0) 0
    (let* ((s0 (sin a0)) (s1 (sin a1))
           (tt (rat (+ (/ s1) (sqrt (1- (/ (* s1 s1)))))
                    (+ (/ s0) (sqrt (1- (/ (* s0 s0))))))))
      (/ 2 (+ tt (/ tt))))))
```

Figure 6. Final algorithm for shortest rational approximations and short sines.

4.8. Applications

The primary advantage of the rational method is that it is distance-preserving (i.e. pure rotations are isometries), and it has an inverse of the same complexity. To illustrate why this is important, we discuss below a fundamental algorithm in robot motion planning where scale-preserving rotations are essential. This is followed by a discussion of two other applications in which our method is practical but not the most efficient solution known. We present them because some algorithms in computational geometry require "rotation" of geometric objects, e.g. to remove degeneracies, or to simplify intersection calculations. Some of these requirements are subtle, e.g. a statement like "Assume the direction of sweep is parallel to the x axis" may actually mask a prescription for rotation. The development of robust geometric algorithms analyzed in the bit-complexity model demands that such prescriptions be made precise. Hence, we would rather see this "rotation" phrased as "Construct the rotation matrix for θ. Approximate the entries with rationals using Euclid's algorithm. Apply the resulting transformation to the environment..." or, "Use the rational method to a choose a rational rotation within ϵ of θ ..." Short of this precision, we provide examples below to show how geometries can be exactly rotated, using the rational method as prescribed by the algorithms. Hence, these examples in a sense demonstrate the correctness of Real-RAM algorithms that prescribe rotation or perturbation of the input, even when implemented using rational arithmetic.

4.8.1. Configuration Space Obstacles

Suppose we wish to compute the configuration space obstacle $CO_A(B)$ for a moving polygon A due to a stationary polygon B (see Lozano-Pérez, 1983). $CO_A(B)$ can be characterized using the Minkowski sum $B \ominus A = \{ b - a \mid a \in A, b \in B \}$. Lozano-Pérez

provided a linear time (optimal) algorithm for the case where A and B are convex, and this algorithm was generalized by Guibas to the non-convex case. In practice, a non-convex A (or B) is often represented as a set of (possibly overlapping) convex polygons; these convex polygons are then pairwise convolved to find a set of configuration space polygons whose union is $CO_A(B)$. This union is often computed using sweep-line techniques. Suppose now that we wish to rotate A by θ and then compute $CO_{A(\theta)}(B)$. $B \ominus A(\theta)$ is exactly a set in which rotated and non-rotated edges must be simultaneously processed. Using rational rotations, the effect on the motion plan would be that the robot rotates to some θ' (the output of our algorithm) within a tolerance ϵ of θ instead. Finally, the computation of $CO_{A(\theta')}(B)$ is exact whereas, for most θ, $CO_{A(\theta)}(B)$ cannot be exactly computed or represented using rational arithmetic.

4.8.2. Rotating the Environment

The following examples involve "rotating" the environment as a pre-process to an algorithm, and then "unrotating" the output of the algorithm. The first example is a line-sweep in an arbitrary direction. The second is a "perturbation" of the environment to remove vertical degeneracies. Both arise frequently in geometric algorithms. The rational method is not optimal for these problems, since it suffices to rotate/perturb the environment by an affine rational matrix that is the concatenation of a rotation and a scaling. Such a matrix can be obtained by simply approximating each entry in the (initial) rotation matrix using Euclid's method (or our optimal version). The topology of the transformed environment is not different, and hence this approximation method would suffice. Nevertheless, we feel it is interesting to see that these operations (problem rotation and perturbation) can be done using pure rational rotation matrices.†

4.8.3. Rotating Line Sweep Problem Instances

To employ the Canny–Donald‡ line sweep projection algorithm in a multistep compliant motion (robot) planner with uncertainty in sensing and control, it is sufficient to compute projections for a finite set of robot velocity angles. For each, we rotate the environment and robot so that the velocity vector points along the negative x axis as shown in figure 7.

The sweep line then moves in the positive x direction and computation of intersections and distances is greatly simplified.§ We can use the rational rotation method presented here to do this rotation of the environment.

4.8.4. Tweaking Line Sweep Problem Instances

The characterization of rational sines by $2/(x + 1/x)$, x rational, has another possible

† A third example involves shortest paths. Suppose we compute the visibility graph G of a set of polygons $A \cup B$. Denote by E_A the (set of) edges of A. Note that as a graph, $E_A \subset G$. Now, compute the visibility graph $G(\theta)$ of $A(\theta) \cup B$. Define $E_A(\theta)$ to be edges of $A(\theta)$, so $E_A(\theta) \subset G(\theta)$. Unless an exact rational rotation method is used, the lengths of the edges in E_A will be *different* from the edges in $E_A(\theta)$. Most disturbing! This application arises in motion planning under rotations.

‡ See Donald (1989, 1990) and Latombe (1990). For readers unfamiliar with this algorithm, the same issues arise in any sweep-line algorithm, e.g. Preparata and Shamos (1985).

§ This is rotating the mountain to Mohammed, if you like.

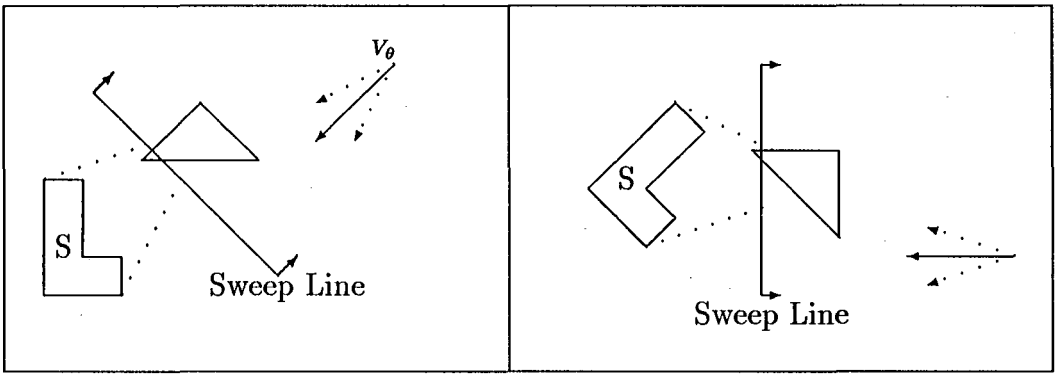

Figure 7. Rotating a projection problem.

application. Many line sweep algorithms in the computational geometry literature are oblivious to the direction of the sweep line. Another common feature is that segments parallel to the sweep line must often be treated as special cases. It might be much simpler and even more efficient to get rid of these cases by rotating the problem so there are no such segments. We can use our new understanding of rational sines to compute short rotation coefficients for this purpose as follows: generate the canonically reduced rationals $0 \leq x \leq 1$ in ascending order of length and pick the first one such that $2/(x + 1/x)$ is the sine for a rotation angle with the desired effect. One approach to generating the x values is to consider the stream of N-mediants of Farey sequences in increasing order of N. Details of an efficient algorithm are left as an exercise.

4.9. Relation to Algebraic Geometry

Rational sines are a special case of a problem that algebraic geometry considers in general. This is the problem of finding rational points on curves. To make this analogy clear, in the case of rational sines, the curve of interest is the unit circle. However, one might also ask for algorithms that produce short rational points on other curves; such questions arise in practice as we discuss below.

Algebraic geometry points to generalizations of this work: equation (4.2) is a genus 0 curve in the (x, y)-plane, which implies it can be parameterized in one variable t. Our characterization of rational sines results from the choice $x = t(y - 1)$ (i.e. substitution in (4.2) yields $y = (t^2 - 1)/(t^2 + 1)$). All quadratic curves have similar parameterizations. This fact yields algorithms for short rational hyperbolic sines and cosines. Note that this technique will not work in general; for example, it would not work for elliptic functions!

5. Organization of this volume

The papers in this volume have been organized into three sections.

5.1. Solving Polynomial Constraints

The first section includes papers on linear and nonlinear polynomial constraints. The papers review continuation methods for determining geometrically isolated solutions of a system of polynomial equations, and then pass to a discussion of recent developments. These constraints arise in many applications, particularly in solid and geometric modeling, machine vision, robotics, kinematics, and design. The papers also review techniques for eliminating variables and computing projections, with an eye towards results we hope may be reduced to practice. A generalization of Sturm's test for determining the roots of a multivariate polynomial is proposed. The papers describe a geometric technique for multivariate projection from a system of linear inequalities.

Morgan's paper is an excellent introduction to homotopy techniques for finding geometrically isolated solutions of a system of polynomial equations. The number of isolated solutions are determined by the Bezout number of a system of polynomial equations, which is the product over the degree of the polynomial equations. The paper reviews the polynomial continuation method, and then discusses many recent developments, including multi-homogeneous polynomial systems, upon which subsystem transformations can lower the Bezout number. It discusses the coefficient-parameter polynomial continuation method. In this technique, coefficients of polynomials are themselves parametric expressions, and so forth; this idea considerably expands the application domains where the continuation methods could be effectively applied. The paper addresses issues encountered in implementing these techniques.

Morgan's paper also discusses the effect of symbolic reduction, e.g. using Gröbner basis methods, on the continuation approach, and warns that a naïve approach to performing symbolic reduction is not likely to win. Using simple examples, the paper demonstrates that the performance of the continuation method on a reduced system may be worse than on the original system.

The paper by Kapur and Lakshman discusses three main approaches for manipulating nonlinear polynomial equations, emphasizing variable elimination and computing projections. It reviews resultant computations for eliminating a single variable, as well as simultaneous elimination of many variables. A particular mention must be made of Dixon's method as well as Macaulay's method. The Gröbner basis algorithm is introduced. This is followed by a discussion of the basis conversion algorithm. This algorithm takes as input a Gröbner basis of a zero-dimensional ideal with respect to one term ordering; it then generates a Gröbner basis with respect to another term ordering. The paper then turns to some recent results of Lazard for constructing a system of triangular sets. Such sets are useful in studying the structure of the zero-set of a system of polynomial equations. Finally, the paper explains Ritt's construction of *characteristic sets*—this construction was recently popularized by Wu in the context of geometric theorem proving. This paper represents a theme in applied computational algebraic geometry; other papers in this area discuss applications of these methods to curve and surface implicitization, detection of "unfaithful" parameterizations, and geometric theorem proving.

Milne's paper proposes a generalization of Sturm's theorem for identifying the isolated zeros of multivariate polynomials (in n variables, say). Around each zero, the method

computes an arbitrarily small n-dimensional box, in which numerical methods can be used to approximate the solution. Instead of computing projections, the algorithm computes the *volume function*, which can be done using elimination techniques such as Gröbner bases, or multivariate resultants.

Milne reports that his technique is based on a generalization of Hermite's method for two dimensions, and is similar to Sturm's method for a single dimension. The isolation phase produces a set of intervals, each containing a single solution; this is done by counting the number of solutions within a given interval and repeatedly subdividing the interval. Once an interval containing a single solution is identified, numerical techniques are used to approximate the solution to a given tolerance. This paper, along with the work of Paul Pederson, is likely to have considerable influence in developing new decision methods for the theory of real closed fields.

Lassez and Lassez's paper proposes a new algorithm for projection (i.e. elimination of certain variables) of a system of linear constraints. Projection is performed geometrically, by first converting the problem into a generalized linear program, and then using a multi-dimensional convex hull algorithm on extreme points. The authors report that the algorithm provides an exact solution when the dimension of output is small, and an approximation in the general case when the output size is very large.

5.2. Machine Vision and Design

The second section includes papers on symbolic and numerical methods in the application domains of machine vision, computational geometry and design.

Ponce and Kriegman's paper is an excellent demonstration of elimination techniques in computer vision. Resultants are used for constructing a model, in the form of an implicit polynomial equations; the indeterminants are the image observables. This model relates observables to the *pose* (i.e. three dimensional position and orientation) of an object model. Implicit equations for models can be derived off-line using resultants. Thus, determining camera parameters (such as pose) becomes a fitting problem between observable data points and the implicit equations defining library object models. Numerical techniques (such as the Lavenberg-Marquardt algorithm for nonlinear, least-square minimization) are used to find a best fit. The approach has been successfully tried on both synthetic and real images of curved objects using range, intensity and contour data. This paper represents a new trend in the field, namely the systematic use of geometric constraints for modeling curved objects, and precise elimination techniques for computing implicit representations of object models.

The paper by Taubin and Cooper proposes an approach using algebraic invariants and covariants, for recognizing and determining the pose of two- and three-dimensional rigid objects. Algebraic surfaces are fitted to regions of image data using numerical techniques. A data base is searched for regions (segments) of library objects having algebraic surface approximations similar to the ones in the image data. Algebraic invariants are used to make the search fast. For each matching candidate (i.e. library objects having algebraic surfaces with approximately the same invariants as the algebraic hypotheses), a coordinate transformation is computed using explicit parametric formulæ. These formulæ are precomputed *a priori* using elimination techniques. Each parametric coordinate transformation represents a covariant function of the coefficients of the polynomials defining a surface. For example: given a nonsingular quadratic surface, the eigenvalues of a "2-

jet"-like matrix† are Euclidean invariants; concomitantly the eigenvectors of the matrix and the center of symmetry of the surface define an intrinsic frame of reference. Finally, the paper discusses methods which generalize these constructions to algebraic curves and surfaces of higher degree.

The paper by Forsyth, Mundy, Zisserman and Rothwell discusses how properties invariant under projective transformations can be used for recognizing planar shapes. The concept of *indexing functions* for describing planar shapes is introduced. Indexing functions have been useful in a fast model-based vision system, which is also described. These concepts are extended to recognize curved, three-dimensional objects. This paper is characteristic of a new direction in machine vision, and demonstrates the effectiveness of coplanar conic invariants for object recognition. It also discusses an approach for determining surface parameters from the outline of an algebraic surface. The paper shows the power of combining techniques such as the symbolic method for computing invariants, elimination methods, and algorithms for functional decomposition, together with numerical techniques like the Lavenberg-Marquardt method (which is used here to solve an induced nonlinear constrained optimization problem to fit the image data).

The paper by Huttenlocher and Kedem discusses distance metrics for comparing planar shapes under different transformations. Having a shape *metric* ensures that if image data matches two library models A and B, then, by the triangle equality, A and B will be very close under the same metric. The failure of many earlier recognition schemes to satisfy the metric properties vitiated development of shape matching theories with powerful predictive power for correctness, completeness, or computational efficiency. This paper emphasizes point-sets in two-dimensional space—sets of line segments and simple polygons in a plane—but the concepts are considerably more general. The effect of translation and similarity transformations (rotation, translation and scaling) on the distance metrics is considered. Efficient algorithms for computing these distance metrics are given. The power of shape metrics lie in (i) having a true metric for shape comparison, (ii) having the metric invariant under groups of image transformations, and (iii) in combinatorial efficiency and precision. These methods are applicable in pattern recognition, machine vision, and robotics.

Murota's paper discusses a mathematical framework for structural analysis of linear dynamical systems. The mathematical model classifies the coefficients of differential equations into independent physical parameters such as mass, spring constants, forces and constant values. These values arise from physical laws (e.g. Kirchoff's laws in electrical systems), laws of physics in mechanical systems, kinematic definitions, and initial conditions. Using this decomposition, the model describes dynamical systems by a class of structured polynomial matrices which are amenable to matroid-theoretic combinatorial analysis. An equivalence relation on matrices is defined using a class of admissible transformations. Each equivalence class has a canonical form, together with an algorithm for computing the canonical form using matroid algorithms and manipulations of bipartite graphs. This approach allows a comprehensive combinatorial analysis of a dynamical system which provides insight into its control-theoretic properties, such as observability and controllability.

The paper by Donald and Pai reviews the use of algebraic methods for simulating mechanical systems with flexible parts, such as those encountered in design for assembly. It is shown that the simulation problem can be reduced to sweeping an arrangement

† This "Hessian-like" matrix is associated with the second-degree terms of the defining polynomials.

of planar algebraic curves of low degree. This avoids both numerical simulation and event-based simulation; furthermore, the simulation is not subject to accumulated errors. The paper illustrates how combinatorial algorithms can be joined with insights from dynamical systems to obtain engineering simulation systems.

5.3. Mathematical Modeling, Symbolic and Numerical Systems

Symbolic algebra systems are often used for modeling and analysis of complex physical phenomena. The output of such an exercise is certain characterizations and properties which can be used to simulate the behavior of a phenomenon using a mathematical model. The third section of this volume includes papers on the mathematical modeling of certain physical phenomena, particularly using partial differential equations. Since a great deal is known about numerical techniques for solving specific classes of partial differential equations, these papers are concerned with preliminary analysis of partial differential equations arising in specific application domains, (i) to set up the parameters required to invoke the appropriate numerical routines, and (ii) to automatically generate code for specific functions called by the numerical routines. For example, a system should be able to compute the Jacobian *symbolically* and then generate efficient *numerical* code for Jacobian; this code must interface with numerical methods for solving PDEs. Other issues dealt with by these papers are: (i) graphic display of results, and (ii) preprocessing to exploit parallel machines.

The three papers—Steinberg's paper, Flaherty, Benantar, Biswas and Moore's paper, and Weerawarana, Houstis and Rice's paper—address how symbolic techniques can be used to enhance the performance of the numerical methods solving partial differential equations that arise in mathematical models.

Steinberg's paper discusses the development of a modeling program for porous media flow in an underground region, using a combination of symbolic and numerical methods. Modeling involves solving an elliptic boundary value problem. An attempt has been made to automate the generation of code needed by numerical algorithms from formulas derived from the symbolic algebra system Macsyma.

The paper by Flaherty, Benantar, Biswas and Moore discusses a software for solving partial differential equations using adaptive techniques. The software provides an interface to Maple, another symbolic algebra system, which facilitates the problem description in a more natural form. Numerical packages that solve PDEs often require the user to supply code for computing the Jacobian of the system; this code is repeatedly called from the package, which uses the results, for example, to bound integration errors. To this end, Maple is used for symbolic manipulation, for example, to preprocess and analyze the symbolic expression for the Jacobian. Code for numerically computing the Jacobian (as well as other functions) is automatically generated. The paper discusses how these adaptive techniques have been implemented on shared-memory parallel computers.

The paper by Weerawarana, Houstis and Rice describes an editor interface to Ellpack and its parallel version //Ellpack, a high level software environment for specifying linear second-order boundary value problems and their solvers. (The "//" means "Parallel".) The editor provides access to Macsyma. The editor can be viewed as a hybrid symbolic-numerical algorithm used to transform PDE problem extensions which cannot be directly solved by Ellpack. Hence, the editor automatically codes the necessary input functions and performs the preprocessing required to generate PDE problems which *can* be solved by Ellpack. For example, symbolic techniques are used for differentiation, whereas

numerical simulation methods are used for solving elliptic PDEs. Linearization is done using Picard or Newton iterations; Fretchet derivatives are generated symbolically. This paper highlights certain obstacles to developing a good hybrid system. The authors point out that software packages are typically designed as closed systems, which makes it very difficult to interface them to other systems. The authors argue that software systems must not only have have a good user interface, but also support a *functionally equivalent* programming interface. Another weakness of many symbolic systems is that it is often impossible to generate, integrate, or manipulate numerical code conveniently within the system.

Zippel's paper discusses a framework for developing special purpose software simulators for mathematical models arising in several different application domains. The environment provides implementations of symbolic and numerical methods, in particular for generating executable code from differential equations. Depending upon the properties of processes being considered, differential equations are simplified (or approximated) into equations which more accurately encode those properties. Numerical integration code is generated using object-oriented techniques in a computer algebra system, WEYL, which extensively uses parameterized and generic structures.

The paper by Davenport, Dewar and Richardson discusses IRENA, which supports an integrated problem-solving environment with symbolic and numerical computation algorithms. IRENA makes the NAG FORTRAN library of numerical algorithms available; the interface is constructed through REDUCE (which is another symbolic algebra system). One chief idea is to spare the user from having to worry about the fiddly formatting requirements of numerical algorithms written in FORTRAN. Second, the output of IRENA makes it easy to present the output from numerical algorithms in an intuitive graphic form. For anyone who has used the NAG library, the benefits of IRENA should be obvious: IRENA provides a simplified interface for accessing the NAG library using the symbolic facilities available in REDUCE. Furthermore, REDUCE users could employ the high-quality numerical algorithms of the NAG library within a REDUCE environment.

References

A. Briggs (1989), "An efficient algorithm for one-step compliant motion planning with uncertainty", *Proc. 5th ACM Symp. Computational Geom.*, Saarbrucken, Germany, 187–196.

R. Brost and M. Mason (1989), "Graphical analysis of planar rigid body dynamics with multiple frictional contacts", *5th Int. Symp. Robotics Research*, Tokyo, 367–374.

F. Bookstein (1979), "Fitting conic sections to scattered data", *Comput. Vision Graph. Image Processing*, **9**, 56–91.

J.F. Canny (1986), "Collision detection for moving polyhedra", *IEEE Trans. Patt. Anal. Mach. Intell.*, **8**(2), 200–209.

J.F. Canny (1989), "On computability of fine motion plans", *IEEE Int. Conf. Robotics and Automation*, Scottsdale, AZ, 177–182.

J.F. Canny, B.R. Donald, J. Reif and P. Xavier (1988), "The complexity of kinodynamic planning", *29th Symp. Foundations Comput. Sci.*, White Plains, NY, 177–183.

J.F. Canny, B.R. Donald and G. Ressler (1992), "A rational rotation method for robust geometric algorithms", *Proc. ACM Symp. Computational Geom.*, Berlin, Germany.

D. Cyrluk, D. Kapur and J.L. Mundy (1987), "Formation of partial 3D models from 2D projections—an application of algebraic reasoning", *DARPA Image Understanding Workshop*, Los Angeles, CA, 798–809.

B.R. Donald (1987), "A search algorithm for motion planning with six degrees of freedom", *Artif. Intell.*, **31**(3), 295–353.

B.R. Donald (1989), *Error Detection and Recovery in Robotics*, Lecture Notes in Comput. Sci. 336, Springer-Verlag, NY.

B.R. Donald (1990), "The complexity of planar compliant motion planning with uncertainty", *Algorithmica*, **5**(3), 353–382.

B.R. Donald and D.K. Pai (1990), "On the motion of compliantly-connected rigid bodies in contact: a system for analyzing designs for assembly", *Proc. IEEE Int. Conf. Robotics and Automation*, Cincinatti, Ohio.

B.R. Donald and P. Xavier (1989), "A provably good approximation algorithm for optimal-time trajectory planning", *IEEE Int. Conf. Robotics and Automation*, Scottsdale, AZ, 958–963.

M. Erdmann (1986), "Using back projections for fine motion planning with uncertainty", *Int. J. Robotics Research*, **5**(1), 19–45.

S.J. Fortune and V. Milenkovic (1991), "Numerical stability of algorithms for line arrangements, *Proc. 7th ACM Symp. Computational Geom.*, New Hampshire, 334–341.

C.M. Hoffmann, J.E. Hopcroft and M.S. Karasick (1988), "Towards implementing robust geometric computations", *Proc. 4th ACM Symp. Computational Geom.*, Urbana, IL, 103–117.

D. Kapur and J.L. Mundy, eds. (1989), *Geometric Reasoning*, MIT Press, Cambridge, MA.

J.-C. Latombe (1990), *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA.

Z. Li and V. Milenkovic (1990), "Constructing strongly convex hulls using exact or rounded arithmetic", *Proc. 6th ACM Symp. Computational Geom.*, Berkeley, CA, 235–243.

T. Lozano–Pérez (1983), "Spatial planning: a configuration space approach", *IEEE Trans. Comput.*, **C-32**, 108–120.

L. Nielson (1988), "Automated guidance of vehicles using vision and projectively invariant marking", *Automatica*, **24**(2), 135–148.

F. Preparata and M. Shamos (1985), *Computational Geometry: An Introduction*, Springer-Verlag, NY.

H. Rademacher (1984), *Lectures on Elementary Number Theory*, Robert E. Krieger, Malabar, Florida.

K. Sugihara (1984), "An algebraic approach to shape from image problems", *Artif. Intell.*, **23**, 59–95.

I. Weiss (1989), "Projective invariants of shapes", *Proc. DARPA Image Understanding Workshop*, Palo Alto, CA, 1125–1134.